The Dual Receiver Cryptosystem and Its Applications

Theodore Diament Homin K. Lee Angelos D. Keromytis Moti Yung
Department of Computer Science, Columbia University
{*tdiament,homin,angelos,moti*}*@cs.columbia.edu*

ABSTRACT

We put forth the notion of a dual receiver cryptosystem and implement it based on bilinear pairings over certain elliptic curve groups. The cryptosystem is simple and efficient yet powerful, as it solves two problems of practical importance whose solutions have proven to be elusive before:
(1) A provably secure "combined" public-key cryptosystem (with a single secret key per user in space-limited environment) where the key is used for both decryption and signing and where encryption can be escrowed and recovered, while the signature capability never leaves its owner. This is an open problem proposed by the work of Haber and Pinkas.
(2) A puzzle is a method for rate-limiting remote users by forcing them to solve a computational task (the puzzle). Puzzles have been based on cryptographic challenges in the past, but the successful design of embedding a useful cryptographic task inside a puzzle, originally posed by Dwork and Naor, remained an open problem till today. We model and present "useful security puzzles" applicable in two scenarios: a secure fileserver, and an online transaction server (such as a webserver).

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems; D.4.6 [**Security and Protection**]: Access Controls

General Terms

Security, Design, Theory

Keywords

Puzzles, Useful Secure Computation, Public Key, Digital Signature, Key Escrow, Pairing-based Cryptography, Elliptic Curves

CCS'04, October 25-29, 2004, Washington, DC, USA.

1. INTRODUCTION

We introduce the notion of a dual receiver cryptosystem, which enables a ciphertext to be decrypted by two independent receivers. To implement a dual receiver cryptosystem, one may use the methodology suggested in [37], by using the first receiver's key and the second receiver's key and encrypting the same plaintext with both. This, however, makes the ciphertext (which should also include a proof of consistent encryption) inefficient. Even in a practice-oriented adaptation of the dual ciphertext methodology [12], such a multi-cryptosystem scheme is not the proper efficient solution.

To achieve a practical system, we build on any bilinear map between two groups, and in particular, we use a pairing defined on certain elliptic curves. Several recent papers have used pairings to construct cryptosystems, and these recent designs inspired our construction's components. The first and most basic design is the three-party one-round Diffie-Hellman key exchange proposed by Joux [28]. A particularly elegant and surprising cryptosystem is the identity-based encryption scheme proposed by Boneh and Franklin [6], in which the public key is a user's identity and a key-generation authority assigns the user a private key. Hierarchical identity-based systems were given in [17], and Boneh, Lynn, and Shacham [7] used pairings to generate short signatures. Various other constructions have been proposed.

Our basic design is a cryptosystem that transforms the three-party one-round Diffie-Hellman key exchange proposed by Joux [28] into a dual receiver cryptosystem public key. This is analogous to the construction of the El Gamal encryption from the Diffie-Hellman key exchange protocol, and is therefore simple and efficient. We show that this simple construction is quite powerful by demonstrating how it solves two open issues in the literature: how to construct public-key cryptosystems that support encryption and signature generation with a single private key per user (called "combined cryptosystems"), and how to construct "useful" computational puzzles secure under certain assumptions.

We begin with an overview of these two problems before we explain the details of the dual receiver cryptosystem. In Section 2, we formally define public-key schemes, the notions of security we used, and the Bilinear Diffie-Hellman assumption, which is the basis for the security of our system. Sections 3 and 4 present the dual receiver ciphertext scheme and its security. Section 5 discusses our decryption-oriented puzzles. The appendices contain an overview of the Tate pairing, and proofs for Theorems 3.1 and 4.1.

1.1 Combined Cryptosystems

Public key systems supporting both encryption and signature generation with a single private key portion per user [21] model a public key in a constrained environment that can afford only a single key per user. The user is required to both sign and decrypt with this single secret key. (This design may arise in constrained environments where a system cannot support enough long term non-volatile memory, or as a flexible design tool in cases where a system is specified to support only signatures but with the outlook that an encryption requirement may be added later after the design is finished). Various systems, *e.g.,* PGP, used combined schemes before, but without any formal proof of security.

The use of a single key for both encryption and signature generation is a problem in certain applications where the encrypted information must be recoverable by a third party (*e.g.,* a security officer). This is true for medical and financial records where privacy and secrecy are becoming mandatory (*e.g.,* the Health Insurance Portability and Accountability Act (HIPAA)), yet the necessity for emergency access remains. However, no authority, and in fact no one but the user, should have the capability to sign the user's name in order to preserve non-repudiation. This issue has been discussed in the literature, as in [13], which recommended that different tasks use separate keys due to escrow. Ideally, the same encryption system should be able to create recoverable ciphertexts for official use and non-recoverable ciphertexts for private use. Even better would be a securely combined encryption and signature system that uses the same private key for both functions, allowing each installation of the system to adopt a "key recovery policy" in a flexible way. The problem of having a *single private key per user* together with simultaneous escrow of the decryption capability and non-escrow of the signing capability has been considered self-contradictory and perhaps somewhat paradoxical, and was one of the earliest criticism raised against combined cryptosystems when they were first suggested [40].

Our work answers this open problem. We propose a combined public key system composed of an encryption scheme secure against chosen-ciphertext attacks in which the encrypted message is either non-recoverable or recoverable by parties that can be chosen differently for each message, and a signature scheme secure against adaptive chosen-message attacks (including attacks by the escrow agent). Both signatures and ciphertexts that use the same secret key consist of a single cryptosystem. Furthermore, a sender can escrow her message to a party of the sender's choosing without this party being involved in any pre-processing. This achieves full flexibility in key management and recovery policy (and the recovering party is only involved in the recovery). Obviously, we do not escrow keys directly in our design but employ the dual receiver cryptosystem method.

As noted by Boneh and Franklin, key escrow is inherent in identity-based schemes, since the key-generation authority knows all the users' private keys. Verheul [46] had suggested that the user have two keys, one that is escrowed to a designated recovery agent who gets involved in key generation and a second one that is not escrowed and can also be used for signing. However, this scheme does not achieve our goals of allowing a single key per user or per server with careful modeling and security proofs. In fact, we are not aware of any prior work that has solved the separation of key management of combined cryptosystems.

1.2 Useful Puzzles

Computational puzzles have been proposed as a means to protect servers from resource-depletion attacks [31, 9, 47], such as TCP SYN floods [45]. The basic idea is to require every client to perform some computationally expensive but easily verifiable (by the server) computation. Attackers issuing large numbers of concurrent requests will need considerable amounts of computational resources, making such attacks difficult to mount, while low-rate legitimate clients will not be severely affected.

Puzzle schemes typically use a one-way function (OWF) to construct a hard computational challenge with a shortcut: knowledge of an input to the OWF allows the server to efficiently compute the result, while a client presented with a result must exhaustively search (brute force) through the space of potential inputs (applying the OWF to each) to determine the correct value. Thus, a server can ask the client a question similar to *"which 32-bit number, when supplied as input to the SHA1 OWF, results in the value 0xdeadbeef"?* The server can pick the input value at random, and may vary its size to reflect the computational resources of the clients and attackers.

However, solving a puzzle represents "useless" computation in the sense that, other than rate-limiting requests, it serves no other purpose. One can imagine a server that uses the clients that request some service to solve a useful problem. Such a "useful puzzle" must have specific properties:
Computational intensity: The puzzle should be a moderate but serious computational task, assuring a certain slowdown of the accessing party (client).
Reliability: It should be computationally efficient for the challenger (server) to verify the result of the puzzle (much easier to check than to compute).
Usefulness: The result of the computation should be useful to the server.
Non-dependability: If the puzzle is not actually solved by the client, the server should still be able to solve it (or give it to another client to solve).
Security: The client must not learn the result of the computation, any long-term cryptographic keys, or any other secrets of the server.

The question is, how can a puzzle be useful, reliable and secure? If it is to be reliable, it means the challenger has to repeat the work of the accessing party in order to verify, thus hurting the notion of "usefulness." If it is to be useful, the accessing party needs some secret information that is required to do the useful work, which may result in giving away the *security* of the challenger (who needs to provide trapdoor information or other secrets that are easier to check). The straightforward approach taken with computational puzzles thus far does not meet these requirements, except for some very specific e-coin minting applications [26]. In this paper, we answer the question on the existence of useful puzzles, first posed in [11], in the affirmative. We propose the first (to our knowledge) "useful security puzzles" scheme where the underlying computation is cryptographic.

At the heart of the solution is the dual receiver cryptosystem. We exploit a cryptosystem that effectively has two receivers: the challenger, who needs to have a puzzle solved as part of a protocol, and a second receiver that is someone who has a different (perhaps temporary) trapdoor that the challenger can provide him with. In our scheme, the second receiver is another client that is contacting the

same challenger (server) and is requesting some service in the same protocol. Getting the trapdoor and doing the useful work is only part of the decryption, since the accessing party should not learn the plaintext, and the result should be easily checkable. What we exploit are padding schemes for chosen-ciphertext secure schemes that separate the trapdoor action from the part that involves checking the integrity of computation and the extraction of plaintexts. The combination of the dual receiver encryption and the separability of the padding scheme enables the useful security puzzles under certain threat models. We show an example whereby the useful task is part of a combined cryptosystem operation.

2. DEFINITIONS

2.1 Public-Key Scheme Definitions

The key management of our scheme enables a second receiver to decrypt the ciphertext. We formally define dual receiver public-key encryption (PKE) schemes, public-key signature schemes, and their combination.

DEFINITION 2.1. **(Dual Receiver Public-Key Encryption Scheme)** *A* dual receiver public-key encryption scheme *consists of four randomized polynomial-time algorithms* $\mathsf{Enc} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ *as follows:*

- *The* key-generation algorithm $\mathcal{K}$ *is a randomized algorithm that takes a security parameter k as an input, and produces a pair (e, d) of corresponding public encryption and private decryption keys. We write $\mathcal{K}(k) = (e, d)$. (Let $\mathcal{K}(k) = (f, g)$ be another key pair in the following.)*
- *The* encryption algorithm $\mathcal{E}$ *is a randomized algorithm that takes public encryption keys e and f, and a message $m \in \mathcal{M}$ (where $\mathcal{M}$ is the message space) as inputs, and produces a ciphertext $c \in \mathcal{C}$ (where $\mathcal{C}$ is the ciphertext space). We write $\mathcal{E}_{e,f}(m) = c$.*
- *The* decryption algorithm $\mathcal{D}$ *is a randomized algorithm that takes a private decryption key d, a public encryption key f, and a ciphertext $c \in \mathcal{C}$ as inputs, and produces a message $m \in \mathcal{M}$ or a special* reject *symbol. We write $\mathcal{D}_{d,f}(c) = m$.*
- *The* recovery algorithm $\mathcal{R}$ *is a randomized algorithm that takes a public encryption key e, a private decryption key g, and a ciphertext $c \in \mathcal{C}$ as inputs, and produces a message $m \in \mathcal{M}$ or a special* reject *symbol. We write $\mathcal{R}_{e,g}(c) = m$.*

We require that if $\mathcal{K}(k)$ outputs (e, d) and (f, g), and $\mathcal{E}_{e,f}(m)$ outputs c, all with positive probability, then $\mathcal{D}_{d,f}(c)$ and $\mathcal{R}_{e,g}(c)$ both output m for all $m \in \mathcal{M}$.

DEFINITION 2.2. **(Public-Key Signature Scheme)** *A* public-key signature scheme *consists of three randomized polynomial-time algorithms* $\mathsf{Sig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ *as follows:*

- *The* key-generation algorithm $\mathcal{K}$ *is a randomized algorithm that takes a security parameter k as an input, and produces a pair (s, v) of corresponding private signature and public verification keys. We write $\mathcal{K}(k) = (s, v)$.*
- *The* signature algorithm $\mathcal{S}$ *is a randomized algorithm that takes a private signature key s and a message $m \in \mathcal{M}$ as inputs, and produces a signature $\sigma \in \{0,1\}^*$. We write $\mathcal{S}_s(m) = \sigma$.*
- *The* verification algorithm $\mathcal{V}$ *is a randomized algorithm that takes a public verification key v and a message-signature pair (m, σ) as inputs, and produces as output either* accept *or* reject. *We write $\mathcal{V}_v(m, \sigma) =$* accept.

We require that if $\mathcal{K}(k)$ outputs (s, v) and $\mathcal{S}_s(m)$ outputs σ, both with positive probability, then $\mathcal{V}_v(m, \sigma) =$ accept, and for any other pair (m, σ'), $\mathcal{V}_v(m, \sigma') =$ reject.

DEFINITION 2.3. **(Combined Scheme)** *Given a dual receiver PKE scheme* $\mathsf{Enc} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D}, \mathcal{R})$ *and a public-key signature scheme* $\mathsf{Sig} = (\mathcal{K}_S, \mathcal{S}, \mathcal{V})$, *the* combined scheme *consists of six randomized polynomial-time algorithms* $\mathsf{Comb} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{R}, \mathcal{S}, \mathcal{V})$ *as follows:*

- *The* key-generation algorithm $\mathcal{K}$ *is a randomized algorithm that takes a security parameter k as an input, and produces two pairs of keys $[(e, d), (s, v)]$, the first for* Enc *and the second for* Sig.
- *Encryption, decryption, and message recovery are performed with $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{R}$, and signature generation and verification are performed with $\mathcal{S}$ and $\mathcal{V}$, exactly as in the original schemes.*

Note that the only differences between the combined and the original schemes are in the key-generation algorithms.

2.2 Security Definitions

We now formally define the security notions we use. Informally, if no probabilistic polynomial-time (PPT) attacker can recover the whole plaintext from a given ciphertext, then the public-key encryption scheme is said to be *one-way*.

DEFINITION 2.4. **(One-Wayness of a Dual Receiver PKE Scheme)** *Given a dual receiver public-key encryption scheme* Enc *and a sufficiently large security parameter k, generate keys $\mathcal{K}(k) = (e, d)$ and $\mathcal{K}(k) = (f, g)$. The success probability of an adversary $\mathcal{A}$,* Succ$(\mathcal{A})$, *is defined to be* $\Pr[\mathcal{A}(\mathcal{E}_{e,f}(m)) = m]$ *where m is a random message in $\mathcal{M}$.* Enc *is (t, ϵ)-*OW, *if for any such adversary $\mathcal{A}$ with running time bounded by $t(k)$,* $\mathrm{Succ}(\mathcal{A}) < \epsilon(k)$.

A *plaintext-checking oracle* takes as input a plaintext m and a ciphertext c and outputs whether or not c encrypts m. If the adversary above has access to a plaintext-checking oracle, it is playing out a *one-way plaintext-checking attack*, or OW-PCA.

Informally, if no PPT attacker can learn any bit of information about the plaintext from the ciphertext, except the length, then a public-key encryption scheme is said to be *semantically secure*, or equivalently *polynomial-time indistinguishable* (notated as IND) [19]. The following definition is the logical extension of semantic security to a dual receiver public-key encryption scheme.

DEFINITION 2.5. **(Semantic Security of a Dual Receiver PKE Scheme)** *Given a dual receiver public-key encryption scheme* Enc *and a sufficiently large security parameter k, generate keys $\mathcal{K}(k) = (e, d)$ and $\mathcal{K}(k) = (f, g)$.*

Given an adversary $\mathcal{A}$ consisting of two PPT algorithms A_1 and A_2, have A_1 choose two equal-length messages (m_0, m_1) from $\mathcal{M}$. For a random bit $b \leftarrow \{0,1\}$, encrypt the corresponding message $\mathcal{E}_{e,f}(m_b) = c$. The advantage of adversary $\mathcal{A}$, $\mathrm{Adv}(\mathcal{A})$, is defined to be $|\Pr[A_2(m_1, m_2, c) = b] - 1/2|$. Enc *is (t,ϵ)-*IND*, or* semantically secure*, if for any such adversary $\mathcal{A}$ with running time bounded by $t(k)$, $\mathrm{Adv}(\mathcal{A}) < \epsilon(k)$.*

The adversary considered above is playing out a *chosen-plaintext attack*, or CPA, since she is able to encrypt any plaintext of her choice. If the adversary has access to both a decryption oracle, and in our case a recovery oracle, then she is playing out a *chosen-ciphertext attack*. Naturally, we do not allow the adversary to ask that m_0 and m_1 be decrypted. If the adversary's access to the oracle is limited in time, the attack is called *non-adaptive* [37]. If access is unlimited, the attack is called *adaptive* or CCA [41]. Chosen-ciphertext security is the strongest security notion that one can expect in the standard model of communication.

When considering signature schemes, if no PPT attacker can forge a signature on one message, then the signature scheme is secure against *existential forgery*.

DEFINITION 2.6. **(Security of a Public-Key Signature Scheme)** *Given* Sig $= (\mathcal{K}, \mathcal{S}, \mathcal{V})$*, a public-key signature scheme, and a sufficiently large security parameter k, generate keys $\mathcal{K}(k) = (s, v)$. Given an adversary $\mathcal{A}$ consisting of a PPT algorithm A, A outputs a message/signature pair, (m, σ). $\mathrm{Adv}(\mathcal{A})$ is defined as $\Pr[\mathcal{V}_v(m, \sigma) =$* accept*$]$.* Sig *is (t,ϵ)-*secure *against* existential forgery *if for any such adversary $\mathcal{A}$ with running time bounded by $t(k)$, $\mathrm{Adv}(\mathcal{A}) < \epsilon(k)$.*

The above adversary is playing out a *key-only attack*, but if the adversary observes valid message/signature pairs chosen and produced by the signer, then she is playing out a *known signature attack*. If the adversary is allowed to ask the signer to sign a number of messages of her choice, then she is playing out a *chosen message attack*. Naturally, we do not allow the adversary to ask that the challenge message be signed. If the adversary's access to the signer is limited in time, the attack is called *non-adaptive*, and if access is unlimited, the attack is called *adaptive* or CMA [20].

When combining a dual receiver public-key encryption scheme Enc with a public-key signature scheme Sig we must verify that sharing keys between the two does not degrade the security of either scheme. Thus, an adversary for Enc with access to a signature-generation oracle should not have a greater probability of success in attacking Enc than it would if it did not have access to the oracle. Similarly, an adversary for Sig with access to a decryption oracle and a recovery oracle should not have a greater probability of success in attacking Sig than it would if it did not have access to either oracle. We prove that the security of a scheme is not degraded in the presence of an oracle by constructing a simulator that does not have the private keys of the scheme, yet can answer the adversary's queries in a manner that is indistinguishable from that of an oracle. If a signature-generation oracle, a decryption oracle, and a recovery oracle can be simulated, then both Enc and Sig can be used in combination without compromising the security of either. Our analysis of the security of the combined scheme uses the technique used by Haber and Pinkas [21] to combine other encryption and signature schemes.

DEFINITION 2.7. **(Security of an Encryption Scheme in a Combined Scheme)** *The combined scheme $\Sigma =$* (Enc, Sig) *does not compromise the security of* Enc *if for any PPT adversary $\mathcal{A}$ with unlimited access to an oracle for $\mathcal{S}_s$, there exists an adversary $\mathcal{A}'$ for* Enc *alone with success probability at most negligibly worse than the success probability of $\mathcal{A}$.*

DEFINITION 2.8. **(Security of a Signature Scheme in a Combined Scheme)** *The combined scheme $\Sigma =$* (Enc, Sig) *does not compromise the security of* Sig *if for any PPT adversary $\mathcal{A}$ with unlimited access to an oracle for $\mathcal{D}_{d,f}$ and $\mathcal{R}_{e,g}$, there exists an adversary $\mathcal{A}'$ for* Sig *alone with success probability at most negligibly worse than the success probability of $\mathcal{A}$.*

DEFINITION 2.9. **(Security of a Combined Scheme)** *The combined scheme $\Sigma =$* (Enc, Sig) *is* CCA-CMA *secure if no PPT adversary $\mathcal{A}$ has a non-negligible advantage against a challenger $\mathcal{A}'$ in a joint* CCA-CMA *game. The adversary is allowed q_1 adaptive queries to signature and decryption oracles, and then picks between a* CCA *or a* CMA *challenge. Once the challenger lays out the challenge, the adversary is allowed q_2 adaptive queries before producing her guess.*

The security model used throughout the paper is that of the random oracle model (a model employed in constructions based on pairings such as [6], and in many efficient constructions for chosen ciphertext secure encryption, *e.g.,* [38]). A random oracle is a function $H : X \rightarrow Y$ chosen uniformly at random from the set of all functions from X to Y, Y finite. An algorithm can query the random oracle for any $x \in X$ and receive $H(x) \in Y$ in response. Random oracles are an idealized model for cryptographic hash functions, and thus security proofs in this model only prove security against attackers that are confined to this model as well. Nevertheless, in many recent designs that employ hash functions as a black box, this design approach followed by a proof in the random oracle model gives a certain validation to the strength of the system design methodology.

2.3 Bilinear Diffie-Hellman Problems

Three related complexity assumptions form the basis of security for cryptography done using discrete logarithms in a group. The security of our encryption scheme is based on the difficulty of the Bilinear Diffie-Hellman Problem, which is an extension of the three problems [10] described below for a multiplicative group $\mathbb{G}$.

DEFINITION 2.10. **(Discrete Logarithm (DL) Problem)** *Given two group elements g and h, find an integer n such that $h = g^n$ whenever such an integer exists.*

DEFINITION 2.11. **(Computational Diffie-Hellman (CDH) Problem)** *Given three group elements g, g^a, and g^b, $a, b \in \mathbb{Z}$, find an element h such that $h = g^{ab}$.*

DEFINITION 2.12. **(CDH Parameter Generator)** *A* CDH parameter generator *$\mathcal{G}$ is a randomized algorithm that takes a security parameter k, and outputs the description of a group $\mathbb{G}$ for which the CDH problem is hard. The CDH problem is considered hard if the following is negligible in k for all PPT algorithms $\mathcal{A}$:*

$$\Pr[\mathbb{G} \leftarrow \mathcal{G}(1^k); g \in \mathbb{G}; a, b \in \mathbb{Z} : \mathcal{A}(\mathbb{G}, g, g^a, g^b) = g^{ab}].$$

$\mathcal{G}$ runs in time polynomial in k, and its order is determined by k.

DEFINITION 2.13. **(Decision Diffie-Hellman (DDH) Problem)** *Given four group elements g, g^a, g^b, g^c with $a, b, c \in \mathbb{Z}$, decide whether or not $c = ab$ (modulo the order of g).*

Note that the DDH problem is no harder than the CDH problem, and that the CDH problem is no harder than the DL problem.

We use a non-degenerate pairing, that is a bilinear map between two abelian groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Let l_1 and l_2 be the orders of the groups. The pairing of two elements $P, Q \in \mathbb{G}_1$ is denoted $\langle P, Q\rangle \in \mathbb{G}_2$. Due to the bilinearity condition, for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, the pair $\langle aP, bQ\rangle = \langle P, Q\rangle^{ab}$. Note that the DL problem should be hard in $\mathbb{G}_2$ so that the pairing is not easily invertible and the DL problem in $\mathbb{G}_1$ is not easily solved [30]. Good candidates for such bilinear maps are the Weil and the Tate pairings defined over points on an elliptic curve defined over a finite field, and they motivate our choice of notating the image in multiplicative notation and the preimage in additive notation.

DEFINITION 2.14. **(Bilinear Diffie-Hellman (BDH) Problem)** *Given the elements P, aP, bP, $cP \in \mathbb{G}_1$ with $a, b, c \in \mathbb{Z}$, find an element $g \in \mathbb{G}_2$ such that $g = \langle P, P\rangle^{abc}$.*

DEFINITION 2.15. **(Decision Bilinear Diffie-Hellman (DBDH) Problem)** *Given the elements P, aP, bP, $cP \in \mathbb{G}_1$ with $a, b, c \in \mathbb{Z}$ and $h \in \mathbb{G}_2$, decide whether or not $h = \langle P, P\rangle^{abc}$. If $h = \langle P, P\rangle^{abc}$, then (P, aP, bP, cP, h) is called a* valid DBDH tuple.

DEFINITION 2.16. **(Gap Bilinear Diffie-Hellman (GBDH) Problem)** *Solve a given instance, (P, aP, bP, cP), of the BDH problem with the help of a DBDH oracle that is able to decide whether or not a tuple $(P, a'P, b'P, c'P, h)$ is valid.*

DEFINITION 2.17. **(BDH Parameter Generator)** *A* BDH parameter generator $\mathcal{G}$ *is a randomized algorithm that takes a security parameter k, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and the description of a non-degenerate bilinear map between the two groups for which the BDH problem is hard. The BDH problem is considered hard if the following is negligible in k for all PPT algorithms $\mathcal{A}$:*

$$\Pr[(\mathbb{G}_1, \mathbb{G}_2, \langle\cdot,\cdot\rangle) \leftarrow \mathcal{G}(1^k); P \in \mathbb{G}_1; a, b, c \in \mathbb{Z} : \mathcal{A}(\mathbb{G}, P, aP, bP, cP) = \langle P, P\rangle^{abc}].$$

Joux [29] gives a detailed analysis of the BDH problem in his survey of the Weil and Tate pairings as building blocks for cryptosystems. The details of the bilinear map used for our cryptosystem will be discussed in Appendix A.

3. THE DUAL RECEIVER CRYPTOSYSTEM

3.1 The Semantically Secure Dual Receiver Scheme

The dual receiver public-key encryption scheme with escrow, SEnc, provides semantic security. The message space is $\mathcal{M} = \{0,1\}^n$. The user may choose any public key yP as the second receiver, and may even choose herself if she does not wish a third-party to have access to the message. Note that the decryption algorithm and the recovery algorithm are the same operations using different keys.

We require that there be a hash function H_x associated with each public key xP; it is easy to base such a family on a given random oracle hash (by first attaching a proper prefix derived from xP to any string to be hashed).

Key-Generation: Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are chosen using a BDH parameter generator $\mathcal{G}$, along with a random element $P \in \mathbb{G}_1$, and $x \in \mathbb{Z}_{l_1}$. The public key is (P, xP) together with a cryptographic hash function $H_x : \mathbb{G}_2 \to \{0,1\}^n$. The private key is x.

Encryption: The input is a plaintext $m \in \{0,1\}^n$. The encryption algorithm chooses a random element $r \in \mathbb{Z}_{l_1}$ and computes $u_1 = rP$, $u_2 = yP$, and $u_3 = m \oplus H_x(\langle xP, yP\rangle^r)$, where H_x is the primary receiver's hash function and yP is the secondary receiver's public key. The ciphertext is (u_1, u_2, u_3).

Decryption: The decryption algorithm for private key x computes $u_3 \oplus H_x(\langle u_1, u_2\rangle^x) = m$. Note that:

$$\langle u_1, u_2\rangle^x = \langle rP, yP\rangle^x = \langle xP, yP\rangle^r = \langle P, P\rangle^{rxy}.$$

Recovery: The recovery algorithm for private key y computes $u_3 \oplus H_x(\langle u_1, xP\rangle^y) = m$. Note that:

$$\langle u_1, xP\rangle^y = \langle rP, xP\rangle^y = \langle xP, yP\rangle^r = \langle P, P\rangle^{rxy}.$$

Security: SEnc is a semantically secure (IND-CPA) dual receiver public-key encryption scheme if the BDH problem is assumed to be hard.

THEOREM 3.1. *Let H_x be a random oracle from $\mathbb{G}_2$ to $\{0,1\}^n$. Let $\mathcal{A}$ be an adversary with running time bounded by t that has advantage ϵ against* SEnc. *Suppose $\mathcal{A}$ makes a total of $q_{H_x} > 0$ queries to H_x. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{G}$ with advantage at least $2\epsilon/q_{H_x}$ and a running time $O(t)$.*

PROOF. See Appendix B. ■

SEnc is also a (OW-PCA) dual receiver public-key encryption scheme if the GBDH problem is assumed to be hard.

LEMMA 3.2. *Let PCO be a plaintext-checking oracle, and let H_x be a random oracle from $\mathbb{G}_2$ to $\{0,1\}^n$. Let $\mathcal{A}$ be an adversary with running time bounded by t that has success probability ϵ against* SEnc. *Suppose $\mathcal{A}$ makes a total of $q_{H_x} > 0$ queries to H_x and PCO. Then there is an algorithm $\mathcal{B}$ that solves the GBDH problem for $\mathcal{G}$ with advantage at least*

$$\frac{\left(\epsilon - \frac{1}{2^n}\right)}{q_{H_x}\left(1 - \frac{1}{2^n}\right)},$$

and a running time $O(t)$.

PROOF. See Appendix C. ■

3.2 The Chosen-Ciphertext Secure Dual Receiver Scheme

The encryption scheme CEnc provides chosen-ciphertext security and allows a specified third party to decrypt the ciphertext. We use the REACT conversion introduced by Okamoto and Pointcheval [38] to convert the SEnc scheme into a chosen-ciphertext secure scheme. The message space is $\mathcal{M} = \{0,1\}^n$, b_2 is the length of the bit-string representation of a point in $\mathbb{G}_2$, and n' is a security parameter.

Key-Generation: Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are chosen using a BDH parameter generator $\mathcal{G}$, as are a random element $P \in \mathbb{G}_1$, and $x \in \mathbb{Z}_{l_1}$. The public key is (P, xP) together with cryptographic hash functions $H_x : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $G : \{0,1\}^n \rightarrow \{0,1\}^n$, and $F : \{0,1\}^{4n+b_2} \rightarrow \{0,1\}^{n'}$. The private key is x.

Encryption: The input is a plaintext $m \in \{0,1\}^n$. The encryption algorithm chooses a random element $r \in \mathbb{Z}_{l_1}$, a random element $\rho \in \{0,1\}^n$, and computes $u_1 = rP$, $u_2 = yP$, $u_3 = \rho \oplus H_x(\langle xP, yP\rangle^r)$, $u_4 = m \oplus G(\rho)$, and $u_5 = F(\rho, m, u_3, u_4, \langle xP, yP\rangle^r)$. The ciphertext is $(u_1, u_2, u_3, u_4, u_5)$.

Decryption: The decryption algorithm computes $u_3 \oplus H_x(\langle u_1, u_2\rangle^x) = \rho$ and $G(\rho) \oplus u_4 = m$ given a ciphertext $(u_1, u_2, u_3, u_4, u_5)$. Then it checks that $u_5 = F(\rho, m, u_3, u_4, \langle u_1, u_2\rangle^x)$, and if u_5 is correct, the algorithm outputs m. Otherwise, it outputs Reject.

Recovery: The recovery algorithm computes $u_3 \oplus H_x(\langle u_1, xP\rangle^y) = \rho$ and $G(\rho) \oplus u_4 = m$, given a ciphertext $(u_1, u_2, u_3, u_4, u_5)$. Then it checks that $u_5 = F(\rho, m, u_3, u_4, \langle u_1, xP\rangle^y)$, and if u_5 is correct, the algorithm outputs m. Otherwise, it outputs Reject.

Security: CEnc is a chosen-ciphertext secure dual receiver public-key encryption scheme if the GBDH problem is assumed to be hard. Since SEnc is OW-PCA and one-time pads (the XORs) are semantically secure, the conversion is chosen-ciphertext secure in the random oracle model. (See Theorem 1 in [38]).

4. THE COMBINED SCHEME

In this section we present a signature scheme that is secure against adaptive chosen-message attacks and the dual receiver encryption scheme that is secure against chosen-ciphertext attacks, both of which use the same private key.

4.1 The Signature Scheme

The signature scheme Sig provides security against existential forgery under a chosen message attack if the CDH problem is assumed to be hard in $\mathbb{G}_1$. The message space is $\mathcal{M} = \{0,1\}^n$. This scheme is similar to Boneh, Lynn, and Shacham's signature scheme [7].

Key Generation: Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are chosen using a BDH parameter generator $\mathcal{G}$, as are a random element $P \in \mathbb{G}_1$, and $x \in \mathbb{Z}_{l_1}$. The public verification key is (P, xP) together with a cryptographic hash function $I : \{0,1\}^n \rightarrow \mathbb{G}_1$. The private signature key is x.

Signature: The input is a private signature key $x \in \mathbb{Z}_{l_1}$ and a plaintext $m \in \{0,1\}^n$. The signature algorithm calculates $\sigma = xI(m)$. The signature is σ.

Verification: Given a public key (P, xP), and a message-signature pair (m, σ), the verification algorithm verifies that $\langle P, \sigma\rangle = \langle xP, I(m)\rangle$.

Security: Sig is secure against existential forgery under adaptive chosen-message attacks.

THEOREM 4.1. *Let I be a random oracle from $\{0,1\}^n$ to $\mathbb{G}_1$. Let $\mathcal{A}$ be an adversary with running time bounded by t that has advantage ϵ against* Sig. *Suppose $\mathcal{A}$ makes a total of $q_I > 0$ queries to I and $q_S > 0$ signature queries. Then there is an algorithm $\mathcal{B}$ that solves the CDH problem for $\mathbb{G}_1$ with advantage at least $\epsilon/e(q_S+1)$ (where e is the base of the natural logarithm) and a running time at most $t + j(q_I + 2q_S)$, where j is the time taken to multiply two points in $\mathcal{G}_1$.*

PROOF. See Appendix D. ■

4.2 The Combined Scheme

Recall that a combined public-key scheme leaves the encryption, decryption, recovery, signature generation, and verification algorithms unchanged, but needs a new key-generation algorithm.

Key-Generation: Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are chosen using a BDH parameter generator $\mathcal{G}$, along with a random element $P \in \mathbb{G}_1$, and $x \in \mathbb{Z}_{l_1}$. The public key is (P, xP) together with a cryptographic hash function $H_x : \mathbb{G}_2 \rightarrow \{0,1\}^n$. The private decryption key is x. The public verification key is (P, xP) together with a cryptographic hash function $I : \{0,1\}^n \rightarrow \mathbb{G}_1$. The private signature key is x.

Security of CEnc in the Presence of Sig: The combined scheme does not compromise the security of *CEnc*.

LEMMA 4.2. *Let I be a random oracle from $\{0,1\}^n$ to $\mathbb{G}_1$. Let $\mathcal{A}$ be an adversary that has advantage ϵ against* CEnc *in a* CCA *attack with unlimited access to I and a signature oracle for* Sig. *Then there is an algorithm $\mathcal{B}$ that has advantage ϵ against* CEnc.

PROOF. Given an adversary $\mathcal{A}$ that attacks CEnc when used together with Sig, we construct an adversary $\mathcal{B}$ attacking CEnc alone.

Algorithm $\mathcal{B}$ is given (P, xP, yP), the encryption key for CEnc, and $\mathcal{B}$ sends the seven-tuple $(P, xP, yP, H_x, G, F, I)$ to $\mathcal{A}$ where I is a random oracle controlled by $\mathcal{B}$.

- *I-queries:* Here I is a random oracle controlled by $\mathcal{B}$ where $\mathcal{B}$ keeps a list of tuples, the I-list. When $\mathcal{A}$ issues a query, q_i, to I, $\mathcal{B}$ checks to see if q_i is on the I-list. If q_i appears in a tuple (q_i, i_i, r_i), then $\mathcal{B}$ responds with $I(q_i) = i_i$. Otherwise, $\mathcal{B}$ picks a random $r_i \in \mathbb{Z}_{l_1}$ and sets $i_i = r_i P$. $\mathcal{B}$ adds the tuple (q_i, i_i, r_i) to the I-list, and responds with $I(q_i) = i_i$.

- *Signature Queries:* When $\mathcal{A}$ issues a signature query, q_i, $\mathcal{B}$ obtains the corresponding tuple, (q_i, i_i, r_i), by making a I-query as outlined above. $\mathcal{B}$ sets $\sigma_i = r_i(xP)$. Note that σ_i is a valid signature for q_i under the public key xP. $\mathcal{B}$ gives σ_i to $\mathcal{A}$.

$\mathcal{A}$'s view of the signature is identical to that of a real signature, and thus its probability of success in breaking the encryption scheme is unchanged. ■

Security of Sig in the Presence of CEnc: The combined scheme does not compromise the security of *Sig*.

LEMMA 4.3. *Let H_x be a random oracle from $\mathbb{G}_2$ to $\{0,1\}^n$, and let F be a random oracle from $\{0,1\}^{4n+b_2}$ to $\{0,1\}^{n'}$. Let $\mathcal{A}$ be an adversary that has advantage ϵ against* Sig *with unlimited access to H_x, F, and a decryption oracle for* CEnc. *Then there is an algorithm $\mathcal{B}$ that has advantage ϵ against* Sig.

PROOF. Given an adversary $\mathcal{A}$ that attacks Sig when used together with CEnc, we construct an adversary $\mathcal{B}$ attacking Sig alone.

Algorithm $\mathcal{B}$ is given the verification key for Sig, (P, xP), and $\mathcal{B}$ sends (P, xP, H_x, G, F) to $\mathcal{A}$ where H_x and F are random oracles controlled by $\mathcal{B}$.

- *H_x-queries:* Here H_x is a random oracle controlled by $\mathcal{B}$ where $\mathcal{B}$ keeps a list of tuples, the H_x-list. When $\mathcal{A}$ issues a query, q_i, to H_x, $\mathcal{B}$ checks to see if q_i is on the H_x-list. If q_i appears in a tuple (q_i, h_i), then $\mathcal{B}$ responds with $H_x(q_i) = h_i$. Otherwise, $\mathcal{B}$ picks a random $h_i \in \{0,1\}^n$, adds the tuple (q_i, h_i) to the H_x-list, and responds with $H_x(q_i) = h_i$.

- *F-queries:* Here F is a random oracle controlled by $\mathcal{B}$ where $\mathcal{B}$ keeps a list of tuples, the F-list. When $\mathcal{A}$ issues a query, q_i, to F, $\mathcal{B}$ checks to see if q_i is on the F-list (note that q_i is a bit-string of length $\{0,1\}^{4n+b_2}$). If q_i appears in a tuple (q_i, r_i), then $\mathcal{B}$ responds with $F(q_i) = r_i$. Otherwise, $\mathcal{B}$ picks a random $r_i \in \{0,1\}^{n'}$, adds the tuple (q_i, r_i) to the F-list, and responds with $F(q_i) = r_i$.

- *Decryption Queries:* $\mathcal{B}$ responds as follows when $\mathcal{A}$ issues the five-tuple decryption query, $q_i = (u_1, u_2, u_3, u_4, u_5)$. $\mathcal{B}$ obtains the corresponding tuple, (q_i, u_5), from the F-list and sets g to the last b_2 bits of q_i. If u_5 is not in any tuple, then $\mathcal{B}$ picks a random $g \in \{0,1\}^{b_2}$. $\mathcal{B}$ then obtains the corresponding tuple (g, h_i) from the H_x list by making a H_x-query as outlined above. $\mathcal{B}$ sets $\rho = u_3 \oplus h_i$, and outputs $m = u_4 \oplus G(\rho)$.

$\mathcal{A}$'s view of the decryption is identical to that of a real decryption (as is its view of the recovery for they are simulated the same way), and thus its probability of success in forging a signature is unchanged. ■

Security of Σ=(CEnc,Sig): The combined scheme Σ is CCA-CMA secure, and thus does not compromise its own security with respect to an adversary that is trying to compromise either the encryption or the signature.

THEOREM 4.4. *Let H_x, F, and I be random oracles, then* Σ *is* CCA-CMA *secure, assuming that the GBDH and the CDH problems are hard.*

PROOF. Queries to H_x, F, I, the decryption and signature oracles are exactly as in Lemmas 4.2 and 4.3. Due to the chosen-ciphertext security of *CEnc*, Theorem 4.1, and Lemmas 4.2 and 4.3, if $\mathcal{A}$ has advantage ϵ over Σ, then there is an algorithm $\mathcal{B}$ that can solve either the GBDH or the CDH problem with non-negligible probability. ■

Non-mandatory Escrow Encryption: Our combined dual receiver encryption and signature scheme can easily allow escrow encryption at the sender's discretion, without any sacrifices to security. The sender can choose a specific escrow public key to be the second key to recover the information. The sender can also choose her own public key to be the second key if she does not wish a third-party to have access to the message. This gives a very flexible key-management component that can be used in designing secure file and storage systems, and general recovery and backup policies. The designation of tasks in the system can be managed via a key certification process.

5. USEFUL SECURITY PUZZLES

Our approach is based on a construct called "decryption-oriented puzzles," which in turn is based on the dual receiver method. We first describe the approach conceptually and then map the implementation to the general approach. We also give an overview of other work in client puzzles, and describe how to use our scheme in two different scenarios: a secure fileserver and an online transaction server.

5.1 Summary of Approach

At an abstract level, we use a "puzzle public key" algorithm for encryption and decryption, where the encryption involves the receiver's public key Ke and an auxiliary key Ka. The public keys are arbitrarily chosen and, given a fixed Ke, any choice for Ka will work. The encryption process generates a ciphertext that has two portions, $(C1, C2)$.

The decryption is such that given one of the private keys Pe or Pa and $(C1, C2)$, the message M can be recovered. We can achieve partial decryption when, given only $C1$ and one of the private keys, an intermediate value, $TD1$, can be found by using a "trapdoor recovery algorithm." The situation is such that: (1) given $C1$, M is unpredictable and remains secure, while; (2) there is a message-recovery algorithm that on $TD1, C1, C2$ can recover the message M quickly. The message recovery algorithm must be much more efficient than the trapdoor recovery algorithm. Further, if a wrong $TD1$ is supplied, the message recovery algorithm will reject it efficiently. This "puzzle public key" algorithm is used as follows:

Consider a busy fileserver where remote clients store and retrieve files over the network. While the communication itself may be protected via a protocol such as TLS, we also want files to be protected while stored on the server's disk. In this scenario, the server has a permanent key Ke as its public key (this may be a combined system where the key is used for encryption and signature), and periodically generates an auxiliary key Ka. When storing a client's file, the server encrypts it using some symmetric cipher (such as AES) under a randomly chosen "session" key K_S, encrypts K_S under the server's puzzle public key (resulting in a $(C1, C2)$ ciphertext per our previous discussion), and stores both the file and the $(C1, C2)$ tuple to disk. Without useful puzzles, the fileserver must decrypt the session key K_S every time a client requests the file. Instead, we want to rate-limit clients requesting files via a puzzle scheme, and lighten the server's computational burden simultaneously.

When a new client (*Client1*) contacts the server with a request for a file, the server responds with the $C1$ portion of that file's encrypted session key, as well as Pa (the auxiliary private key). *Client1* will be able to solve the puzzle $(C1)$ using that key and give the result, which is the trapdoor $TD1$, to the server. The server will try to recover the message via a very efficient computation. If successful, *Client1* will get access (since it solved the puzzle and extracted the trapdoor). If the message recovery fails (*e.g.,* because *Client1* did not solve the puzzle), *Client1* will be denied access. The knowledge of the auxiliary private keys and the trapdoors has no effect on security, since the session keys cannot be recovered by clients and the auxiliary keys are relatively short–lived. Furthermore, the server can now provide the decrypted file to *Client1*. A malicious attacker could retrieve a collection of files legitimately, memorize their trapdoors, and then reuse them to flood the server with repeated requests for those files, avoiding having to spend the time to solve the puzzle. We are concerned with the computationally expensive aspect of this process, the session key decryption. Other mechanisms (*e.g.,* caching) must used to avoid other attacks.

A second scenario of using decryption puzzles involves an online transaction server (such as a webserver) to which clients connect using a session-security protocol similar to TLS. Here, clients encrypt the session keys that will protect the remainder of the session under the server's puzzle public key. We discuss this scenario in more detail in Section 5.4, as the security model is somewhat different from the relatively straightforward case of the fileserver.

5.2 Decryption-Oriented Puzzles

We present a construction of a decryption-oriented puzzle from any semantically secure dual-receiver PKE such as the one presented in Section 3.1. We use the REACT transform introduced by Okamoto and Pointcheval [38] (employing strong cryptographic hash functions that behave like a random oracle) to build a system that is chosen-ciphertext secure and has the desired puzzle properties.

Let $\mathsf{Enc}=(\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{R})$ be as in Definition 2.1 with message space $\mathcal{M}=\{0,1\}^n$.

- *Server Key-Generation:* The key generation algorithm $\mathcal{K}$ is run on security parameter k and produces a pair $\mathcal{K}(k)=(e,d)$. The public key is e together with two cryptographic hash functions, $G:\{0,1\}^n \rightarrow \{0,1\}^n$ and $F:\{0,1\}^{4n} \rightarrow \{0,1\}^{n'}$. The private key is d.
- *Auxiliary Key-Generation (for Client):* The key generation algorithm $\mathcal{K}$ is run (separately and with independent coins) on security parameter k and produces a pair $\mathcal{K}(k)=(f,g)$. The public auxiliary key is f. The private auxiliary key is g.
- *Encryption:* The input is a plaintext message $m \in \{0,1\}^n$. The encryption algorithm chooses a random string $\rho \in \{0,1\}^n$ and applies the dual-receiver encryption $\mathcal{E}$ with public keys e,f to produce an encryption $c=\mathcal{E}_{e,f}(\rho)$. It also computes $u_1=m\oplus G(\rho)$ and $u_2=F(\rho,m,c,u_1)$. The ciphertext is $(C_1=c;C_2=[u_1,u_2])$.
- *Trapdoor Decryption by Client:* Given a ciphertext $C_1=c$, the first part of the of the decryption algorithm applies the dual-receiver recovery algorithm $\mathcal{R}_{e,g}(c)$ to recover $\rho=TD1$.
- *Decryption by the server:* Given a ciphertext $(C_1=c;C_2=[u_1,u_2])$ and the trapdoor $TD1=\rho$, the message decryption algorithm computes $G(\rho)\oplus u_1=m$ and it checks that $u_2=F(\rho,m,c,u_1)$, and if u_2 is correct, the algorithm outputs m. Otherwise it outputs Reject. If it rejects, the server can compute ρ itself from $C_1=c$ and repeat this step with the new ρ (checking the sender's integrity in this case).

We can show, and briefly argue that:

LEMMA 5.1. *The decryption–oriented puzzle scheme satisfies* Usefulness, Security, Integrity and Recovery.

PROOF. *Usefulness:* Note that while the trapdoor recovery, which results in ρ, gives only a random value, it involves performing the costly (computationally intensive) dual-receiver message recovery algorithm $\mathcal{R}$. In our implementation (Section 3.1) this is an operation over the curve (the pairing) and is conceptually analogous to exponentiation over a finite field, if somewhat more complex in terms of the actual operations involved. On the other hand, the message decryption by the server involves only bit-wise XOR, and a check of a simple hash function (which is much more efficient). Thus the puzzle is very useful yet reliable.

Security: The scheme is a chosen-ciphertext secure public-key encryption. The client, seeing only part of the ciphertext that recovers to a random value ρ, has no idea what the message is (he only sees parts that could have been computed without seeing the message).

Integrity and Recovery: The check in REACT assures the faithful operation by the client. On the other hand, if the operation fails to pass verification, the server can do the puzzle itself from scratch, or give it to another client. ■

5.3 Related Work on Puzzles

Early work defending against resource depletion attacks focused around the concept of the "cookie," an opaque bit-string that the initiator of a connection request needs to return verbatim to the server before the request is allowed to proceed. Thus, cookies were used only to establish the validity of the peer in terms of network address reachability; in other words, cookies protect against attackers spoofing their IP address. Cookie-based solutions [36] were used against TCP connection-depletion (also known as TCP SYN) attacks [45, 23], and in security protocols such as Photuris [32], IKE [22], JFK [2], and others [39, 24]. More generally, The advantages of being stateless, at least in the beginning of a protocol run, were recognized in the context of security protocols in [27] and [4].

Computational client puzzles as a means to defend against denial of service attacks were first introduced in [31]. In that work, client puzzles were used to counter TCP SYN attacks from attackers that were willing to expose their IP address. Although TCP cookies are ineffective in that scenario, client puzzles can mitigate the effects of such an attack by an adversary that is CPU-limited. However, in recent years attackers have demonstrated their ability to effectively utilize large numbers of subverted hosts in their attacks [25]. Gligor [18] argues that solutions requiring client proofs of work (*e.g.,* computational client puzzles such as those using hash functions) are both ineffective and unnecessary in open networks, such as the Internet, when strong access guarantees (*e.g.,* maximum waiting time) are desired.

Jakobsson and Juels [26] first proposed the concept of a useful puzzle, which they call a "bread pudding protocol." The particular scheme they use, applied to minting e-coins for the MicroMint micropayment system [42], is specific to MicroMint and does not appear to be easily generalizable to other types of useful work.

Client puzzles have also been used in the context of security protocols [26, 35], most notably for protecting SSL against computational denial of service attacks [9]. Other uses of client puzzles involve junk email mitigation [11], fair exchange [8, 16], protection of sensor networks against DoS attacks [48], and time-lock puzzles [43]. The latter aims to encrypt a message such that it cannot be decrypted, even by the sender, until some pre-determined future time. A summary of other uses of client puzzles (also known as "hash cash") may be found in [5]. [1] introduced the concept of a memory-bound puzzle, which aims to impose the same solving delay as traditional client puzzles by increasing the number of memory accesses a client needs to perform to solve the challenge.

Wang and Reiter [47] introduce the idea of a puzzle auction as a way to ease some of the practical deployment difficulties, *e.g.,* selecting the appropriate hardness for the puzzles. Their approach lets clients bid for the resources by adjusting the difficulty of the puzzles they solve. When the server is attacked, legitimate clients gradually increase their bids (puzzle difficulty), eventually bringing the cost outside the adversary's capabilities.

5.4 Employing Useful Security Puzzles in a TLS-like Protocol

We now briefly discuss another application of useful puzzles, which trades off security against certain kinds of eavesdropping adversaries with resistance to computational denial of service attacks. Consider a cryptographic protocol such as TLS, a simplified version of which is shown in Figure 1 (inspired by a similar figure in [33]). In Message 3 of this protocol, the client encrypts a randomly-chosen secret value, S, with the server's public key (obtained from the certificate sent by the server in Message 2). The server must decrypt this secret value, which both parties use to derive a session key. In almost all cases, the RSA algorithm is used to encrypt S. Note that the server also uses its private key to authenticate (via a signature) to the client.

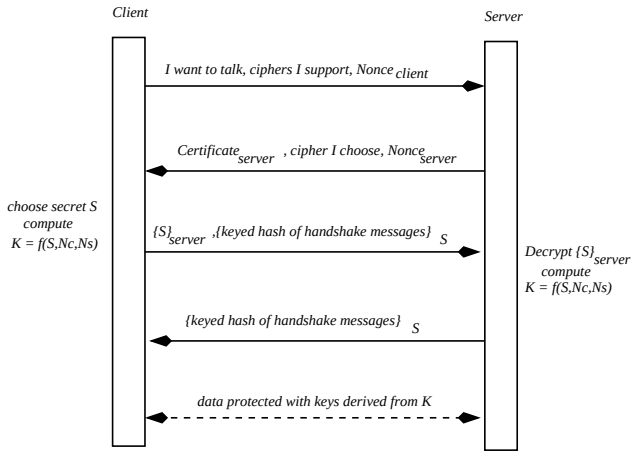

Figure 1: Simplified SSLv3/TLS.

We envision using useful security puzzles as a substitute for the RSA encryption shown above, as shown in Figure 2. The server will have a long-term public/private key pair (Ke and Pe), and will periodically select a new auxiliary key pair (Ka and Pa). A client A that contacts the server will receive both public keys (Ka and Ke), as well as the current auxiliary private key (Pa). The client will then select a secret S, which will be encrypted along with a server-provided stateless nonce Ns, using both public keys creating ciphertext $C1, C2$, as described in Section 5.1. If the server is lightly loaded, it may simply decrypt this value using Pa and Pe itself. Otherwise, the server selects another client, B, at random and forwards $C1$ to it. On a busy server, such as a popular e-commerce web site, there will be a constant stream of new clients connecting, to which $C1$ can be forwarded to. Similarly, the original client may receive another $C1'$, produced by another client connecting to the server.

Client B will now use Pa to produce the intermediate value $TD1$ and send it back to the server as proof of work done. The server will verify the solution (as described in Section 5.2) and the nonce, and will allow the connection from client B to proceed. At the same time, the server has retrieved the secret value S produced by client A for use in deriving a session key. The purpose of the nonce is to force colluding clients (*e.g.,* machines controlled by the same attacker) to communicate with each other, mitigating the impact of an influx of such clients on the throttling properties of our scheme. If client A also provides a correct response to the challenge $C1'$ (which may have been generated by client B or some other client contacting the server in the same window of time as A), it will be allowed to proceed with its connection. Neither A nor B have learned anything about the secret values they helped decrypt, nor have they learned anything that would allow them to impersonate the server to other clients (*e.g.,* the server's private key). The server has throttled down the clients by forcing them to perform some useful computation; under schemes such as [31], the client would have to perform the same work in addition to solving a "useless" puzzle, while the server itself would have to do more of the protocol's cryptographic work.

There is, however, a weakness in the use of the puzzle in this environment: a powerful adversary that is capable of monitoring all the server's communication links can obtain enough information (specifically, $(C1, C2)$ and $TD1$) to decrypt the original message from the client to the server, thus violating the security of the TLS-like protocol. From a denial-of-service perspective, such an adversary can potentially perform much more powerful attacks (*e.g.,* shut down these links); however, this scheme has the potential to make things worse from a security perspective. Addressing this issue remains an open problem; here, we offer two potential approaches to mitigating the threat.

First, we can treat the first iteration of the TLS-like protocol as a pre-authentication phase, establishing a key which can be used to quickly validate the client's traffic to the server's router using a scheme such as the one proposed by Yaar, Perrig and Song [49]; a second authentication phase (without using puzzles) is subsequently used to secure the end-to-end path. Second, we can use a distributed set of servers through which the main server routes (and receives) Messages 3, 4 and 5 (per Figure 2). These messages are transmitted under pre-established security associations, preventing an attacker eavesdropping on the server's direct links from obtaining enough information. Such an attacker would instead need to eavesdrop the links for all these servers. Recent work has shown that such overlay-based mechanisms offer reasonable security guarantees [34] and performance characteristics [3].

6. CONCLUSIONS

We introduced the notion of a "dual receiver cryptosystem" which enables a ciphertext to be decrypted by two independent receivers. We presented a construction and illustrated its use in two important applications that solve heretofore open problems in the literature.

The first application, inspired by the work of Haber and Pinkas, is a "combined cryptosystem" wherein multiple participants, each maintaining only a single public/private-key pair, can both encrypt and sign messages, and can also delegate decryption (escrow) capabilities to a specified user (on a per-message basis, if desired). The escrow is achieved with-

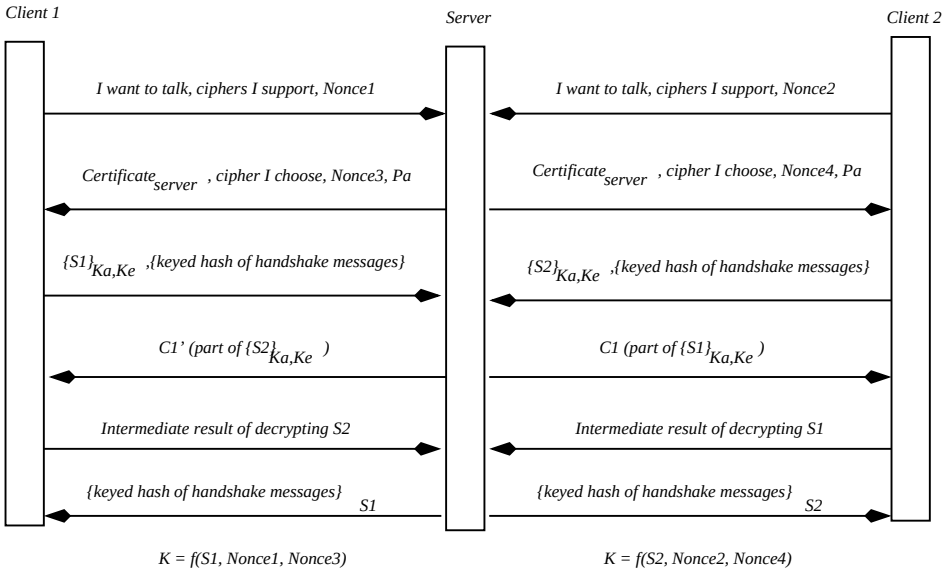

Figure 2: A TLS-like protocol using useful puzzles. For simplicity, only two clients are shown, each partially decrypting the other's secret value S on behalf of the server.

out compromising the security of the signature scheme or the security of any other message encryption.

The second application, first suggested by Dwork and Naor, is a "decryption-oriented" useful puzzle scheme. Here, a server can effectively delegate the decryption of an encrypted message to a client in the form of a puzzle. The puzzle-solving client facilitates the decryption without learning anything about the encrypted message or the server's private key. The remaining cryptographic workload for the server (including verification that the puzzle was correctly solved) is reduced to a bitwise XOR and the computation of a simple hash. We believe that this scheme will have important applications in preventing denial-of-service attacks, and we explore two such settings: a network fileserver and a web server. The happy irony is that a DoS attacker that seeks to shut down a server by inducing it to perform computationally intensive cryptographic computations, is forced to facilitate the server's pending cryptographic tasks on behalf of legitimate clients.

7. REFERENCES

[1] M. Abadi, M. Burrow, M. Manasse, and T. Wobber. Moderately Hard, Memory-bound Functions. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, February 2003.

[2] B. Aiello, S. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. Keromytis, and O. Reingold. Efficient, dos-resistant secure key exchange for internet protocols. In *ACM Computers and Communications Security conference (CCS)*, 2002.

[3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP)*, October 2001.

[4] T. Aura and P. Nikander. Stateless connections. In *Proceedings of International Conferenec on Information and Communications Security (ICICS), Lecture Notes in Computer Science volume 1334*, pages 87–97. Springer, November 1997.

[5] A. Back. Hashcash - A Denial of Service Counter-Measure. `http://www.cypherspace.org/hashcash/hashcash.pdf`, August 2002.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[8] D. Boneh and M. Naor. Timed Commitments (Extended Abstract). In *Proceedings of CRYPTO*, pages 236–254, August 2000.

[9] D. Dean and A. Stubblefield. Using Client Puzzles to Protect TLS. In *Proceedings of the 10th USENIX UNIX Security Symposium*, August 2001.

[10] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, IT-22(6):644–654, Nov. 1976.

[11] C. Dwork and M. Naor. Pricing via Processing, or Combating Junk Mail. In *Proceedings of CRYPTO*, pages 139–147, August 1992.

[12] P. Fouque and D. Pointcheval. Thrshold cryptosystems secure against chosen-ciphertext attacks. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 351 – 368. Springer-Verlag, 2001.

[13] Y. Frankel and M. Yung. Escrow encryption systems visited: attacks, analysis and designs. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 222–235. Springer-Verlag, 1995.

[14] G. Frey, M. Müller, and H.-G. Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[15] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D. R. Kohel, editors, *Proc. Algorithmic Number Theory, 5th International*

Symposium (ANTS-V), volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.

[16] J. A. Garay and M. Jakobsson. Timed Release of Standard Digital Signatures. In *Proceedings of the 6th Conference on Financial Cryptography*, pages 168–182, February 2002.

[17] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer-Verlag, 2002.

[18] V. D. Gligor. Guaranteeing Access in Spite of Distributed Service-Flooding Attacks. In *Proceedings of the Security Protocols Workshop*, April 2003.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.

[20] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[21] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In P. Samarti, editor, *Proc. 8th ACM Conference on Computer and Communications Security*, pages 215–224. ACM Press, 2001.

[22] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force, Nov. 1998.

[23] L. Heberlein and M. Bishop. Attack Class: Address Spoofing. In *Proceedings of the 19th National Information Systems Security Conference*, pages 371–377, October 1996.

[24] S. Hirose and K. Matsuura. Enhancing the resistance of a provably secure key agreement protocol to a denial-of-service attack. In *Proceedings of the 2nd International Conference on Information and Communication Security (ICICS)*, pages 169–182, November 1999.

[25] K. Houle, G. Weaver, N. Long, and R. Thomas. Trends in Denial of Service Attack Technology. `http://www.cert.org/archive/pdf/DoS_trends.pdf`, October 2001.

[26] M. Jakobsson and A. Juels. Proofs of Work and Bread Pudding Protocols. In *Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security*, September 1999.

[27] P. Janson, G. Tsudik, and M. Yung. Scalability and flexibility in authentication services: the KryptoKnight approach. In *Proceedings of IEEE INFOCOM*, pages 725–736, April 1997.

[28] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proc. Algorithmic Number Theory, 4th International Symposium (ANTS-IV)*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[29] A. Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In C. Fieker and D. R. Kohel, editors, *Proc. Algorithmic Number Theory, 5th International Symposium (ANTS-V)*, volume 2369 of *Lecture Notes in Computer Science*, pages 20–32. Springer-Verlag, 2002.

[30] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Manuscript. Available from `eprint.iacr.org`, 2001.

[31] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 151–165, February 1999.

[32] P. Karn and W. Simpson. Photuris: Session-key management protocol. Request for Comments (Experimental) 2522, Internet Engineering Task Force, Mar. 1999.

[33] C. Kaufman, R. Perlman, and M. Speciner. *Network Security, 2nd Edition*. Prentice Hall, 2002.

[34] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.

[35] J. Leiwo, P. Nikander, and T. Aura. Towards network denial of service resistant protocols. In *Proceedings of the 15th International Information Security Conference (IFIP/SEC)*, August 2000.

[36] J. Lemmon. Resisting SYN-flood DoS Attacks with a SYN Cache. In *Proceedings of the USENIX BSD Conference (BSDCon)*, February 2001.

[37] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*, volume 547 of *Lecture Notes in Computer Science*, pages 427–437. Springer-Verlag, 1990.

[38] T. Okamoto and D. Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In B. Preneel, editor, *Topics in Cryptology — CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2002.

[39] R. Oppliger. Protecting key exchange and management protocols against resource clogging attacks. In *Proceedings of the IFIP TC6 and TC11 Joint Working Conference on Communications and Multimedia Security (CMS)*, pages 163–175, September 1999.

[40] B. Pinkas. Personal communication.

[41] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1991.

[42] R. Rivest and A. Shamir. PayWord and MicroMint. *CryptoBytes*, 2(1):7–11, 1996.

[43] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock Puzzles and Timed-release Crypto. Technical Report MIT/LCS/TR-684, MIT, 1996.

[44] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptography. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 336–353. Springer-Verlag, 2002.

[45] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni. Analysis of a Denial of Service Attack on TCP. In *IEEE Security and Privacy Conference*, pages 208–223, May 1997.

[46] E. R. Verheul. Evidence that XTR is more secure than supersingluar elliptic curve cryptosystems. In B. Pfizmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer-Verlag, 2001.

[47] X. Wang and M. K. Reiter. Defending Against Denial-of-Service Attacks with Puzzle Auctions (Extended Abstract). In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2003.

[48] A. D. Wood and J. A. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer*, 35(10):54–62, Oct. 2002.

[49] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Security and Privacy Symposium*, May 2004.

APPENDIX

A. THE PAIRING

Recall that our scheme makes use of a non-degenerate pairing, that is a bilinear map between two groups $\mathbb{G}_1$ and $\mathbb{G}_2$. We choose $\mathbb{G}_1$ to be a large subgroup of the group of points of an elliptic curve over $\mathbb{F}_q$ of order $l \neq q$, where $q = p^n$. $\mathbb{G}_2$ is chosen to be a subgroup of $\mathbb{F}_{q^r}^*$ where r is the security multiplier and $q^r - 1$ is divisible by l.

Two different pairings can be defined over an elliptic curve – the Weil pairing and the Tate pairing. We give preference to the Tate pairing as it is faster to compute [15]. Another issue is that a low security multiplier is needed for the pairing to be efficiently computed, and r always reaches its optimal value in the Tate pairing, but does not always do so for the Weil pairing [14].

Background: Before defining the Tate pairing we review some definitions.

Let $k(x, y)$ denote the field of quotients (rational functions in x, y with coefficients in the field k), where $f = g/h$ and g and h are homogeneous of the same degree. Let the function field of the curve E, denoted $k(E)$, be the equivalence classes of rational functions on E, $k(E) = k(x, y)/I(E)$. If $f = g/h \in k(E)$, then $h \notin I(E)$, and two functions g/h and g'/h' are identified if $gh' = g'h$.

Let $k[E] = k[x, y]/I(E)$ be the coordinate ring of E (where the quotient field is $k(E)$ from before). We define the ideal M_P of $k[E]$ by $M_P = \{f \in k[E] : f(P) = 0\}$. Given $f \in k[E]$, the *order* of f at point P is denoted $\mathrm{ord}_P(f)$, and is the multiplicity of the point. We can also define $\mathrm{ord}_P(f)$ as $\max\{d \in \mathbb{Z} : f \in M_P^d\}$. Using $\mathrm{ord}(f/g) = \mathrm{ord}_P(f) - \mathrm{ord}_P(g)$, we can extend ord_P to $k(E)$. Given $f \in k(E)$, we say that f has a *zero* at $P \in E$ if $\mathrm{ord}_P(f) > 0$, and a *pole* at P if $\mathrm{ord}_P(f) < 0$. Equivalently, f has a zero at P if $f(P) = 0$, and a pole at P if $f(P)$ is not finite, denoted $f(P) = \infty$.

The *divisor group* of a curve E, denoted $\mathrm{Div}(E)$ is the free Abelian group generated by the points of E. Thus a divisor $D \in \mathrm{Div}(E)$ is a formal sum, $\sum_{P \in E} n_P(P)$, with $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many $P \in E$. The *degree of* D is defined by $\deg D = \sum_{P \in E} n_P$. The *support* of D is the set of points for which $n_P \neq 0$.

Given $f \in k(E)$, we can associate a divisor $\mathrm{div}(f)$ to f, given by $\mathrm{div}(f) = \sum_{P \in E} ord_P(f)(P)$. A divisor D is *principal* if it has the form $D = \mathrm{div}(f)$ for some $f \in k(E)$ (f is unique up to constant multiples). Two divisors D_1 and D_2 are *linearly equivalent*, denoted $D_1 \sim D_2$ if $D_1 - D_2$ is principal. Given $f, g \in k(E)$, functions such that div(f) and div(g) have disjoint support, *Weil's Reciprocity Law* states that $f(\mathrm{div}(g)) = g(\mathrm{div}(f))$.

An *elliptic curve* is a pair (E, O), where E is a cubic curve in two variables and $O \in E$ is the point at infinity (the elliptic curve is usually just referred to as E). The elliptic curve E is *defined over* k, denoted E/k, if E is defined over k as a curve and $O \in E(k)$. Given an elliptic curve E and $D \in \mathrm{Div}(E)$, D is principal if and only if $\deg D = 0$ and $\sum n_P(P) = O$. Given a function $f \in k(E)$ and a divisor $D = \sum n_P(P)$, f can be evaluated at D by defining $f(D) = \prod f(P)^{n_P}$ for P in the support of D.

Tate pairing: Given $l \in \mathbb{Z}$, $l \neq 0$, the *l-torsion subgroup* of E, denoted $E[l]$, is the set of points of order l in E. $E[l] = \{P \in E : lP = O\}$. Given an l-torsion point P, D_P denotes a divisor from the class $(P) - (O)$ of the quotient of group of divisors of degree 0 by the subgroup of principal divisors, and f_P denotes a function such that $\mathrm{div}(f_P) = l(P) - l(O)$. The *Tate pairing* of two points $P, Q \in E[l]$ is defined as $t(P, Q) = f_P(D_Q)^{(q^r-1)/l}$. Recall that $l|(q^r - 1)$.

An elliptic curve defined over a field of characteristic q is *supersingular* if $E[q^r] = 0$ for all $r \geq 1$. Supersingular curves have extra endomorphisms in their endomorphism ring, and these endomorphisms map points defined over the ground field to points defined over an extension field. Thus, given an endomorphism Φ, $t(P, \Phi(P)) \neq 1$, and we do not have to worry about the points in the pairing being linearly independent. We denote $t(P, \Phi(Q))$ by $\hat{t}(P, Q)$.

Hashing: Recall that the signature generation scheme requires a cryptographic hash function $I : \{0, 1\}^n \to \mathbb{G}_1$. To construct such a hash function we first construct a hash function from $\{0, 1\}^n$ to $\mathbb{F}_q$, and then we construct an encoding function from $\mathbb{F}_q$ to $\mathbb{G}_1$. Given $M \in \{0, 1\}^n$, M is hashed to $f \in \{0, 1\}^{\log q}$ and it is rehashed if it not less than q. Let y be the f-th element of $\mathbb{F}_q$. Given y, the encoding function calculates x from the equation for E, and sets $P = (x, y)$. The encoding function outputs lP, which is an element in $\mathbb{G}_1$.

Extensions: Our system can use the Weil pairing, as well as pairings over more general Abelian varieties [44]. More general bilinear maps of the form $m : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_2$ can also be used. In this case, both ciphertext and signature can be shortened in length by taking $\mathbb{G}_0$ to be a subgroup of $\mathbb{F}_p$ and $\mathbb{G}_1$ to be a different subgroup of $\mathbb{F}_{p^6}$ of the same order. Both the Weil and the Tate pairings can be used on the asymmetric pair $\mathbb{G}_0 \times \mathbb{G}_1$ as the map m. See [7] for details.

B. PROOF OF THEOREM 3.1

PROOF. The proof of this theorem closely follows Lemma 4.3 in [6]. Algorithm $\mathcal{B}$ is given the BDH parameters produced by $\mathcal{G}$ and an instance of the BDH problem for these parameters, (P, aP, bP, cP). $\mathcal{B}$ uses the adversary $\mathcal{A}$ to find $g = \langle P, P\rangle^{abc}$, the solution to the BDH problem, as follows. First, $\mathcal{B}$ creates a public key for SEnc by setting $xP = aP$ and $yP = bP$, and sends (xP, yP, H_x) to $\mathcal{A}$.

- *H_x-queries:* Here H_x is a random oracle controlled by $\mathcal{B}$ where $\mathcal{B}$ keeps a list of pairs, the H_x-list. When $\mathcal{A}$ issues a query, q_i, to H_x, $\mathcal{B}$ checks to see if q_i is on the H_x-list. If q_i appears in a pair (q_i, h_i), then $\mathcal{B}$ responds with $H_x(q_i) = h_i$. Otherwise, $\mathcal{B}$ picks a random string $h_i \leftarrow \{0, 1\}^n$, adds the pair (q_i, h_i) to the H_x-list, and responds with $H_x(q_i) = h_i$.
- *Challenge:* The adversary $\mathcal{A}$ produces two messages m_0 and m_1 on which it wishes to be challenged. $\mathcal{B}$ picks a random string $u \in \{0, 1\}^n$, defines the ciphertext to be (cP, u), and gives the ciphertext as the challenge to $\mathcal{A}$. Note that the decryption of the ciphertext is $u \oplus H_x(\langle aP, bP\rangle^c) = u \oplus H_x(g)$ due to the way we defined xP and yP.
- *Guess:* The adversary $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. $\mathcal{B}$ responds by outputting a random q_i that appears on the H_x-list as the solution to the given instance of the BDH problem.

Let Q be the event that $\mathcal{A}$ issues a query for g. Then $\Pr[Q]$ in the simulation is the same as $\Pr[Q]$ in the real attack. Before any queries are made $\Pr[Q] = 0$ in both cases. Let Q_i be the event that a query for g was made in the first i queries. $\Pr[Q_i] = \Pr[Q_i|Q_{i-1}] \Pr[Q_{i-1}] + \Pr[Q_i|\neg Q_{i-1}] \Pr[\neg Q_{i-1}]$, and using induction we only have to show that $\Pr[Q_i|\neg Q_{i-1}]$ in the simulation is the same as $\Pr[Q_i|\neg Q_{i-1}]$ in the real attack. Note that the public key and the challenge are distributed as in the real attack, and all responses to the H_x-queries are uniform and independent in $\{0, 1\}^n$. Thus, $\Pr[Q_i|\neg Q_{i-1}]$ is the same in both the simulation and the real attack, and $\Pr[Q]$ is the same in both the simulation and the real attack.

If $\mathcal{A}$ never issues a query for g, then the decryption of the ciphertext is independent of $\mathcal{A}$'s view. Therefore in the real attack $\Pr[b = b'|\neg Q] = 1/2$. Since $\mathcal{A}$ has advantage ϵ, $|\Pr[b = b'] - 1/2| \geq \epsilon$.

$$\begin{aligned}\Pr[b = b'] &= \Pr[b = b'|\neg Q]\Pr[\neg Q] + \Pr[b = b'|Q]\Pr[Q] \\ &\leq \frac{1}{2}\Pr[\neg Q] + \Pr[Q] = \frac{1}{2} + \frac{1}{2}\Pr[Q].\end{aligned}$$

Therefore $\epsilon \leq |\Pr[b = b'] - 1/2| \leq 1/2\Pr[Q]$, and $\Pr[Q] \geq 2\epsilon$. The probability that g appears in some pair on the H_x-list is at least 2ϵ, and thus $\mathcal{B}$ produces the correct answer with probability at least $2\epsilon/q_{H_x}$. ■

C. PROOF OF LEMMA 3.2

PROOF. Algorithm $\mathcal{B}$ is given the BDH parameters produced by $\mathcal{G}$ and an instance of the GBDH problem for these parameters, (P, aP, bP, cP). $\mathcal{B}$ uses the adversary $\mathcal{A}$ and a DBDH oracle to find $g = \langle P, P\rangle^{abc}$, the solution to the GBDH problem, as follows. First, $\mathcal{B}$ creates a public key for SEnc by setting $xP = aP$ and $yP = bP$, and sends (xP, yP, H_x) to $\mathcal{A}$.

- *H_x-queries:* H_x-queries are handled as they are in Theorem 3.1.
- *PCO-queries:* Here PCO is a plaintext-checking oracle controlled by $\mathcal{B}$. PCO-queries are equivalent to reverse H_x-queries. When $\mathcal{A}$ issues a query, $(m_i, c_i = (u_{1_i}, u_{2_i}, u_{3_i}))$, to PCO, $\mathcal{B}$ checks to see if $h_i = m_i \oplus u_{3_i}$ is on the H_x-list. If h_i does not appear in a pair (q_i, h_i), then $\mathcal{B}$ picks a random element $r \leftarrow \mathbb{Z}_{l_2}$, sets $q_i = \langle P, P\rangle^r$, and adds the pair (q_i, h_i) to the H_x-list. $\mathcal{B}$ uses a DBDH oracle to determine if $(xP, u_{1_i}, u_{2_i}, q_i)$ is a valid DBDH tuple. If it is, $\mathcal{B}$ responds YES to $\mathcal{A}$, and otherwise responds NO.
- *Challenge:* $\mathcal{B}$ picks a random string $u \in \{0,1\}^n$, defines the ciphertext to be (cP, yP, u), and gives the ciphertext as the challenge to $\mathcal{A}$. Note that the decryption of the ciphertext is $u \oplus H_x(\langle aP, bP\rangle^c) = u \oplus H_x(g)$ due to the way we defined xP and yP.
- *Guess:* The adversary $\mathcal{A}$ outputs its guess $m \in \{0,1\}^n$. $\mathcal{B}$ responds by outputting a random q_i that appears on the H_x-list as the solution to the given instance of the GBDH problem.

Let Q be the event that $\mathcal{A}$ issues a query for g in a H_x-query or a query for $H_x(g)$ (actually a query for some $(m', (xP, yP, m' \oplus H_x(g))))$ in a PCO-query. Then $\Pr[Q]$ in the simulation is the same as $\Pr[Q]$ in the real attack. Before any queries are made $\Pr[Q] = 0$ in both cases. Let Q_i be the event that a query for g or $H_x(g)$ was made in the first i queries. All the responses by the PCO-queries are valid and only create entries in the H_x-list that are uniform and independent in $\{0,1\}^n$. Using similar reasoning to that used in the proof for Theorem 3.1, $\Pr[Q_i|\neg Q_{i-1}]$ is the same in both the simulation and the real attack, and thus $\Pr[Q]$ is the same in both the simulation and the real attack.

Let $S \stackrel{\text{def}}{=} \Pr[\mathcal{A}((cP, yP, u))]$. If $\mathcal{A}$ never issues a query for g or $H_x(g)$, then the decryption of the ciphertext is independent of $\mathcal{A}$'s view. Therefore in the real attack $\Pr[S|\neg Q] = 1/2^n$. Since $\mathcal{A}$ has success probability ϵ, $\Pr[S] \geq \epsilon$.

$$\begin{aligned}\Pr[S] &= \Pr[S|\neg Q]\Pr[\neg Q] + \Pr[S|Q]\Pr[Q] \\ &\leq \frac{1}{2^n}\Pr[\neg Q] + \Pr[Q] = \frac{1}{2^n} + \left(1 - \frac{1}{2^n}\right)\Pr[Q].\end{aligned}$$

Therefore $\epsilon \leq \Pr[S] \leq \frac{1}{2^n} + \left(1 - \frac{1}{2^n}\right)\Pr[Q]$, and

$$\Pr[Q] \geq \frac{\epsilon - \frac{1}{2^n}}{1 - \frac{1}{2^n}}.$$

The probability that g or H_x appears in some pair on the H_x-list is at least $\left(\epsilon - \frac{1}{2^n}\right) / \left(1 - \frac{1}{2^n}\right)$, and thus $\mathcal{B}$ produces the correct answer with probability at least

$$\frac{\left(\epsilon - \frac{1}{2^n}\right)}{q_{H_x}\left(1 - \frac{1}{2^n}\right)}$$

. ■

D. PROOF OF THEOREM 4.1

PROOF. Algorithm $\mathcal{B}$ is given the CDH parameters produced by $\mathcal{G}$ and an instance of the CDH problem for these parameters, (P, aP, bP). $\mathcal{B}$ uses the adversary $\mathcal{A}$ to find $g = abP$, the solution to the CDH problem, as follows. First, $\mathcal{B}$ creates a verification key for Sig by setting $xP = (a + r)P$ for a random $r \in \mathbb{Z}$, and sends (P, xP, I) to $\mathcal{A}$.

- *I-queries:* Here I is a random oracle controlled by $\mathcal{B}$ where $\mathcal{B}$ keeps a list of tuples, the I-list. When $\mathcal{A}$ issues a query, q_i, to I, $\mathcal{B}$ checks to see if q_i is on the I-list. If q_i appears in a tuple (q_i, i_i, r_i, c_i), then $\mathcal{B}$ responds with $I(q_i) = i_i$. Otherwise, $\mathcal{B}$ picks a random $r_i \in \mathbb{Z}$ and generates a random coin $c_i \in \{0,1\}$ where $\Pr[c_i = 0] = 1/(q_S + 1)$. If $c_i = 0$, $\mathcal{B}$ sets $i_i = r_iP + bP$. If $c_i = 1$, $\mathcal{B}$ sets $i_i = r_iP$. $\mathcal{B}$ adds the tuple (q_i, i_i, r_i, c_i) to the I-list, and responds with $I(q_i) = i_i$.
- *Signature Queries:* When $\mathcal{A}$ issues a signature query, q_i, $\mathcal{B}$ obtains the corresponding tuple by making a I-query as outlined above. If $c_i = 0$, $\mathcal{B}$ reports failure and terminates. If $c_i = 1$, $\mathcal{B}$ sets $\sigma_i = r_i(aP) + ri_i = (a+r)r_iP$. Note that σ_i is a valid signature for q_i under the public key xP which was set to $(a + r)P$. $\mathcal{B}$ gives σ_i to $\mathcal{A}$.
- *Challenge:* The adversary $\mathcal{A}$ produces m, σ, a message-signature pair on which it wishes to be challenged such that m was never a signature query. If σ is not a valid signature on m, $\mathcal{B}$ reports failure and terminates. Otherwise, $\mathcal{B}$ obtains the corresponding tuple by making a I-query as outlined above. If $c_i = 1$, then $\mathcal{B}$ reports failure and terminates. Otherwise, $c_i = 0$ and $i_i = r_iP + bP$. Thus, $\sigma = (a + r)(r_i + b)P$. Then $\mathcal{B}$ outputs the required solution to the CDH problem as $abP = \sigma/(rbP + r_iaP + r_irP)$.

Let Q_1 be the event that $\mathcal{B}$ does not abort during $\mathcal{A}$'s signature queries. The probability that $\mathcal{B}$ does not abort during one query is $1 - 1/(q_S + 1)$, and since $\mathcal{A}$ makes at most q_S signature queries, $\Pr[Q_1] \geq (1 - 1/(q_S + 1))^{q_S} \geq 1/e$.

Let Q_2 be the event that $\mathcal{A}$ produces a valid message-signature pair given that the Challenge stage was successfully reached. The public key given to $\mathcal{A}$ is from the same distribution as a public key produced by the key-generation algorithm, and the responses to the I-queries are uniformly and independently distributed in $\mathbb{G}_1$. Thus $\Pr[Q_2] \geq \epsilon$.

Let Q_3 be the event that the final I-query made by $\mathcal{B}$ does not fail given that $\mathcal{B}$ did not report a failure up to this point. The probability that $c = 0$ is $1/(q_s + 1)$ so $\Pr[Q_3] \geq 1/(q_s + 1)$.

If Q_1, Q_2, and Q_3 are true then $\mathcal{B}$ produces the correct answer. Thus $\mathcal{B}$ solves the CDH problem with probability

at least $\epsilon/e(q_s + 1)$. $\mathcal{B}$'s running time is the time it takes for $\mathcal{A}$ to run plus the time it takes to respond to $(q_I + q_S)$ hash queries and q_S signature queries. If a multiplication in $\mathbb{G}_1$ takes time j, then the total running time is at most $t + j(q_I + 2q_S)$. ■