Authenticated Identity-Based Encryption

Ben Lynn
blynn@cs.stanford.edu

June 4, 2002

Abstract

Suppose Alice wishes to send a message to Bob using an identity-based encryption scheme (recall such a scheme is a public key cryptosystem where any string is a valid public key), but desires integrity as well as security. In other words, Alice wants Bob to know that only she could have sent the message. Furthermore, suppose she does not want the non-repudiation property that would necessarily be present if she simply used an identity-based signature scheme i.e. she does not want Bob to be able to prove to a third party that she is the sender.

We augment the system of Boneh and Franklin [2] to allow communication with integrity without nonrepudiation. We formalize notions of security and integrity for our scheme, and show that new encryption and decryption algorithms are more efficient, despite being equally secure and authenticated.

1 Introduction

When people converse with one another in private, they enjoy secure, authenticated communication that is not non-repudiable. For example, if Alice is whispering to Bob, Bob knows that Alice is transmitting a message securely to him, but is later unable to prove any fact about that message to a third party, including the fact that Alice even sent a message to him. These features are often exactly what the sender of a message desires.

Thus it is natural to ask for these traits when communicating electronically. Ideally such a scheme should be noninteractve so that it applies to email, or other stateless protocols. Many systems providing secrecy exist, but authenticity is usually achieved by signing messages, which is non-repudiable; the receiver can now prove to a third party that the sender sent a particular message.

With standard (non-identity-based) cryptosystems, one can use designated verifier signature schemes [10] to remove non-repudiation, or alternatively, integrate authentication with encryption by using traditional public-key cryptosystem constructs [14]. However, these systems require heavy use of certificates, and in some settings, an identity-based system would be preferable.

Recall that an identity-based encryption (IBE) system is a public-key cryptosystem where any string is a valid public key. Identity-based encryption and signature schemes were asked for in 1984 [12]. Soon after, various identity-based signature schemes were proposed [6, 7] but a fully-functional identity-based identity encryption scheme was not found until recently by Boneh and Franklin [2].

We present a method for integrating authentication with encryption in the Boneh-Franklin IBE system. The changes are such that the authenticated encryption and decryption algorithms are faster than the corresponding non-authenticated (anonymous) versions. It is based on an idea independently discovered by Sakai, Ohgishi and Kasahara [11].

Authenticated encryption is highly desirable in IBE email systems. Another application for authenticated IBE systems is in the construction of a stateless secure network protocol [1].

We shall see that an authenticated IBE system is readily obtained by extending the system of Boneh and Franklin, and integrity can be had for no extra cost. In fact, authenticated encryption is more efficient. Interestingly, the message authentication code is the ciphertext itself, thus proving integrity is equivalent to showing ciphertext unforgeability.

2 Identity-Based Encryption

An *identity-based encryption scheme* is consists of the following four algorithms [2]: Setup, Extract Encrypt, and Decrypt. In summary, Setup generates publicly distributed system parameters and a master key, Extract generates private keys corresponding to a given primitive ID, Encrypt encrypts a message using a given ID, and Decrypt decrypts a ciphertext given a private key. We shall always take the message space to be $\mathcal{M} = \{0,1\}^*$ unless otherwise specified.

We require these algorithms to satisfy the standard consistency constraint, namely when d is the private key generated by algorithm Extract when it is given the IDA as the public key, then

$$\forall M \in \mathcal{M} : \mathsf{Decrypt}(\mathsf{params}, A, C, d) = M \text{ where } C = \mathsf{Encrypt}(\mathsf{params}, A, M).$$

We define authenticated encryption and decryption algorithms for IBE schemes:

Authenticated-Encrypt: input: a message, a private key (the sender's), and an ID (the receiver's), output: a ciphertext.

Authenticated-Decrypt: input: a ciphertext, an ID (the sender's), a private key (the receiver's), output: the corresponding plaintext.

We require these two algorithms to satisfy the standard consistency constraint. Additionally, for all messages M, and for any two ID's A, B with corresponding private keys d_A, d_B, we require $\mathsf{Authenticated\text{-}Encrypt}(M, d_A, B) = \mathsf{Authenticated\text{-}Encrypt}(M, A, d_B)$.

3 Formalizing Security and Integrity

In our scheme, an adversary is able to forge or decrypt a message if they know the sender's or the receiver's private key, so our models do not allow the adversary access to these keys. Briefly, the adversary is allowed to do practically anything but directly obtain information about the sender's or receiver's private keys.

3.1 Security

Consider the following game, played by two parties: an adversary and a challenger.

1. The challenger runs the Setup algorithm for a given security parameter k and gives the system parameters to the adversary. It does not divulge the master key.

2. For an arbitrary number of rounds, the adversary can submit one of the following queries:

Encryption query: the adversary submits any two ID's (sender and receiver) and any plaintext, and is told the resulting ciphertext.

Decryption query: the adversary submits any two ID's and any ciphertext, and is told the corresponding plaintext.

Key generation query: the adversary submits any ID, and is told the corresponding private key.

The queries may be adaptive (i.e. each query may depend on the replies to previous queries).

3. The adversary then outputs any two plaintexts $M_0, M_1 \in \mathcal{M}$ and any two ID's A, B on which it wishes to be challenged on, subject to the restriction that the private keys of A, B have not been queried in the previous step.

4. The challenger picks $b \in \{0, 1\}$ randomly and computes $C = \mathsf{Authenticated\text{-}Encrypt}(\mathsf{params}, d_A, B, M_b)$, where d_A is the private key of A. It sends the challenge C to the adversary.

5. The adversary again issues some number of encryption, decryption and/or key generation queries adaptively, except now it may not ask for the keys of A, B or the plaintext corresponding to C.

6. The adversary outputs $b' \in \{0, 1\}$, and wins if $b = b'$.

Such an adversary is called an AID-CCA attacker.

An authenticated identity-based encryption scheme to be secure against adaptive chosen ciphertext attack (AID-CCA) if no polynomially-bounded adversary has a non-negligible advantage in the above game, that is, for any probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathrm{Adv}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right|$ is less than $1/f(k)$ for all polynomials f. (The probability is over the random bits used by the two parties.)

3.2 Integrity

For integrity, since the ciphertext is the MAC of the message in our scheme, we require ciphertext unforgeability.

Consider the following game, played by two parties: an adversary and a challenger.

1. The challenger runs the Setup algorithm for a given security parameter k and gives the system parameters to the adversary.

2. The adversary submits some number of encryption, decryption and/or key generation queries adaptively.

3. The adversary then attempts to output any valid ciphertext C from a sender A to a receiver B, provided it has not queried the private keys of A, B in the previous step. The adversary wins if the ciphertext is valid.

Call such an adversary an AID-CUF attacker. We say an IBE scheme is secure against ciphertext forgery (AID-CUF) if no polynomially-bounded adversary has a non-negligible advantage in the above game.

4 Extending the Boneh-Franklin IBE Scheme

4.1 The Boneh-Franklin IBE Scheme

We briefly review the IBE scheme given by Boneh and Franklin [2].

Definition 4.1 *Let G_1, G_2 be groups with prime order q. Then we say a map e: $G_1 \times G_1 \to G_2$ is* bilinear *if for all $g, h \in G_1$ and $a, b \in \mathbb{F}_q$, we have $e\left(g^a, h^b\right) = e(g,h)^{ab}$.*

Definition 4.2 *The* Bilinear-Diffie-Hellman problem *(BDH) for a bilinear function e: $G_1 \times G_1 \to G_2$ such that $|G_1| = |G_2| = q$ is prime is defined as follows: given $g, g^a, g^b, g^c \in G_1$, compute $e(g,g)^{abc}$, where g is a generator and a, b, c are randomly chosen from $\mathbb{F}_q$. An algorithm is said to solve the BDH problem with advantage ε if*

$$\Pr\left[\mathcal{A}\left(g, g^a, g^b, g^c\right) = e(g,g)^{abc}\right] \geq \varepsilon.$$

Definition 4.3 *A randomized algorithm $\mathcal{IG}$ that takes as input a security parameter $k \in \mathbb{Z}$ (in unary) is a* BDH parameter generator *[3] if it runs in time polynomial in k and outputs the description of two groups G_1, G_2 and a bilinear function e: $G_1 \times G_1 \to G_2$, with $|G_1| = |G_2| = q$ for some prime q. Denote the output of the algorithm by $(G_1, G_2, e) = \mathcal{IG}\left(1^k\right)$.*

Definition 4.4 *We say that $\mathcal{IG}$* satisfies the BDH assumption *if no probabilistic polynomial-time algorithm $\mathcal{A}$ can solve BDH (for $\mathcal{IG}\left(1^k\right)$) with non-negligible advantage.*

We use the same Setup and Extract algorithms as the Boneh-Franklin scheme except that we require an additional hash function. The original Encrypt and Decrypt algorithms are not authenticated, though they are worth retaining for encrypting when no private key is available, or for anonymous encryption.

Setup: input: $k \in \mathbb{Z}$. Run $\mathcal{IG}\left(1^k\right)$. Choose a random $a \in \mathbb{F}_q$ and a random $g \in G_1$. Pick cryptographic hash functions H_1: $\mathbb{F}_q \times G_2 \to \{0,1\}^n$, H_2: $\{0,1\}^* \to G_1$, H_3: $\{0,1\}^* \times \{0,1\}^* \to \mathbb{F}_q$, H_4: $\{0,1\}^n \to \{0,1\}^n$, (for some n). For the security proof, we view the hash functions as random oracles.

Output: the master key a and params $:= \langle \mathcal{IG}\left(1^k\right), g, g^a, H_1, H_2, H_3, H_4 \rangle$.

Extract: input: an IDA, output: $d_A = H_2(A)^a$.

4.2 Authenticated Encryption

We add the following algorithms to the system. Suppose we have a semantically secure symmetric cryptosystem. We represent its encryption and decryption functions with a key K by E_K, D_K, and assume the keyspace is $K \in \{0,1\}^*$.

Authenticated-Encrypt: input: a mesage $M \in \{0,1\}^*$, a private key d_A, an ID B, and the system parameters. Choose a random $\sigma \stackrel{R}{\leftarrow} \{0,1\}^n$, compute $r = H_3(\sigma, M)$ and $s := e(d_A, H_2(B))$ Then output the ciphertext $C := \langle r, \sigma \oplus H_1(r,s), E_{H_4(\sigma)}(M) \rangle$

Authenticated-Decrypt: input: a ciphertext $\langle U, V, W \rangle$, an ID A, a private key d_B. Compute $s := e(H_2(A), d_B)$, $\sigma := V \oplus H_1(U,s)$, $M := D_{H_4(\sigma)}(W)$

Check that $U = H_3(\sigma, M)$. If not, reject the ciphertext, otherwise output then plaintext M.

Consistency is clear since $e(d_A, H_2(B) = e(H_2(A), H_2(B))^a = e(H_2(A), d_B)$ by bilinearity.

Note that Authenticated-Encryptis faster than plain Encrypt because there is one less exponentiation and no point multiplication.

Note also that now both algorithms benefit greatly from caching: if Alice and Bob expect to send many messages to each other they can both compute s once and cache the result, obviating the need for an expensive Weil pairing computation during encryption and decryption which makes their communication as fast as a symmetric cipher and MAC. (In the original system, caching helped encryption but not decryption.)

4.3 Proof of Integrity

Theorem 4.1 *Suppose $\mathcal{A}$ is a polynomially-bounded attacker that can forge a ciphertext with advantage ε and makes at most Q H_2 queries and at most Q_D decryption queries. Then there exists a polynomially-bounded algorithm $\mathcal{B}$ that solves the BDH problem with advantage at least $\varepsilon / Q_D \binom{Q}{2}^2$.*

Proof The algorithm $\mathcal{B}$ is given g, g^a, g^b, g^c, and its goal is to output $e(g,g)^{abc}$. It uses $\mathcal{A}$ as a subroutine. There is a list L_2 that stores information on H_2 queries, a list L_1 for H_1 queries, and a list of possible answers L_s. All lists are initially empty.

$\mathcal{B}$ runs $\mathcal{A}$ giving it the system parameters g, g^a. Then two random numbers i, j between 1 and Q are chosen. Without loss of generality we may assume that all H_2 queries are distinct (as the replies can be cached), and that if a query involving an $\mathsf{ID}A$ is issued (be it encryption, decryption or key generation) then $\mathcal{A}$ has already issued an H_2 query for A.

There are several assumptions we may make about $\mathcal{A}$'s behaviour when interacting with the decryption oracle. Firstly, we may assume before $\mathcal{A}$ gives its guess, $\mathcal{A}$ issues a decryption query on it. Next we may assume $\mathcal{A}$ does not issue decryption queries on ciphertexts it has received from the encryption oracle, or ciphertexts it can compute because it has previously asked for the private key of the sender or receiver. Lastly, given these assumptions, we may assume that after every decryption query on a ciphertext, if the reply is a plaintext (i.e. the ciphertext it queried is valid) then $\mathcal{A}$ stops and outputs this ciphertext, because it has clearly won.

An H_2 query on an $\mathsf{ID}A$ is handled thus:

1. If it is the ith query, respond with g^b. We call A a guessed ID.
2. If it is the jth query, respond with g^c. We call A a guessed ID.
3. Otherwise, choose a random $d \xleftarrow{\mathrm{R}} \mathbb{F}_q$, insert the tuple $\langle A, d \rangle$ into L_2 and return g^d.

An H_1 query on a tuple $\langle r, h \rangle \in \mathbb{F}_q \times G_2$ is handled thus: a random $R \xleftarrow{\mathrm{R}} \{0,1\}^n$ is output, and the tuple $\langle r, h, R \rangle$ is inserted into the list L_1.

Next, $\mathcal{A}$'s queries of step 2 are handled:

> **Encryption**: suppose $\mathcal{A}$ issues an encryption query for a plaintext M between ID's A and B. If A and B are the guessed ID's, then B picks a random $R \xleftarrow{\mathrm{R}} \{0,1\}^n$, a random $r \xleftarrow{\mathrm{R}} \mathbb{F}_q$ and a random $K \xleftarrow{\mathrm{R}} \{0,1\}^n$. Then $\mathcal{B}$ outputs the ciphertext $C := \langle r, R, E_K(M) \rangle$. (It turns out that the message that is encrypted can be arbitrary, but for simplicity we fix it to be equal to the queried plaintext.)
>
> Otherwise without loss of generality assume A is not a guessed ID. By assumption the list L_2 must contain the entry $\langle A, d \rangle$ for some $d \in \mathbb{F}_q$. Then A's private key is g^{ad}, and the ciphertext is computed as described by the Authenticated-Encryptalgorithm. (i.e. $s := e(H_2(B), g^{ad})$ is computed and so on.) $\mathcal{A}$ is given the ciphertext.

Decryption: suppose $\mathcal{A}$ issues an decryption query for a ciphertext ciphertext $C = \langle U, V, W \rangle$ between ID's A and B. If A and B are the guessed ID's, then L_1 is examined for an entry of the form $\langle U, s, R \rangle$ for some s, R. If such an entry is present s is added to the list L_s. $\mathcal{A}$ is notified that C is invalid, even if C is valid.

Otherwise without loss of generality assume A is not a guessed ID. Again the list L_2 must contain the entry $\langle A, d \rangle$ for some $d \in \mathbb{F}_q$, and so g^{ad} is A's private key. Then the ciphertext is decrypted as outlined in the description of the Authenticated-Decryptalgorithm. If valid, the plaintext given to $\mathcal{A}$ (and $\mathcal{A}$ wins).

Key generation: suppose $\mathcal{A}$ issues a key generation query for an IDA. If A is a guessed IDthen $\mathcal{B}$ fails. Otherwise the list L_2 must contain the entry $\langle A, d \rangle$ for some d and $\mathcal{B}$ outputs g^{ad}.

Eventually $\mathcal{A}$ terminates. Any output is ignored. Now if L_s is empty then $\mathcal{B}$ fails, otherwise $\mathcal{B}$ outputs a random element of L_s.

Firstly, the probability that $\mathcal{A}$ never issues a key generation query on one of the guessed ID's, is at least $1/\binom{Q}{2}$ (there are at least two ID's it cannot ask the keys for).

If $\mathcal{A}$ has submitted a valid ciphertext, then with at least probability $1/\binom{Q}{2}$, $\mathcal{A}$ has successfully forged a ciphertext between the guessed ID's (but is told that the ciphertext is invalid).

In this case, it is evident that if $s = e(g,g)^{abc}$ is not on the list L_s, then $\mathcal{A}$'s view is independent of a correct forgery, because we are modeling H_1 as a random oracle. Thus the probability that $\mathcal{A}$ queries $H_1(s)$ is at least ε. If this happens, then $\mathcal{B}$ cannot fail because L_s is not empty, and outputs the correct s with probability at least $1/Q_D$ (as the size of the list is bounded by Q_D).

It is possible for $\mathcal{A}$ to distinguish between this simulation and real life:

1. If $\mathcal{A}$ asks for an encryption of a particular message between the guessed ID's, it may be able to tell that the simulation's ciphertext is invalid (which will be the case with overwhelming probability). However, because H_1 and H_4 are modeled as random oracles, this is only possible if $\mathcal{A}$ has made a H_1 query on $s = e(g,g)^{abc}$, in which case s will appear on L_s.

2. If $\mathcal{A}$ queries $H_1(U, s)$ for any U and $s = e(g,g)^{abc}$, and has also issued a decryption query for $C = \langle U, V, W \rangle$ for some valid ciphertext C on the guessed ID's, it may realize it is being lied to when it is told that C is invalid. However, again this implies $s = e(g,g)^{abc}$ will have been recorded on L_s.

In other words, $\mathcal{A}$ can realize that it is in a simulation only after it has deposited $e(g,g)^{abc}$ on L_s.

Otherwise, because we are modeling the hash functions as random oracles, we are guaranteed that $\mathcal{A}$ cannot tell the difference (with overwhelming probability) between the simulation and real life.

4.4 Proof of Security

Theorem 4.2 *Suppose $\mathcal{A}$ is a polynomially-bounded AID-CCA attacker with advantage ε whose number of H_2 queries is bounded by Q, and whose number of H_1 queries is bounded by Q_1. Furthermore, suppose our scheme is ciphertext unforgeable. Then there exists a polynomially-bounded algorithm $\mathcal{B}$ that can solve the BDH problem with advantage $\varepsilon / Q_1 \binom{Q}{2}^2$.*

Proof The algorithm $\mathcal{B}$ is given g, g^a, g^b, g^c, and its goal is to output $e(g,g)^{abc}$. It uses $\mathcal{A}$ as a subroutine. There is a list L_2 that stores information on H_2 queries, a list L_1 for H_1 queries, and a list of possible answers L_s. All lists are initially empty. $\mathcal{B}$ runs $\mathcal{A}$ giving it the system parameters g, g^a.

Then two random numbers i, j between 1 and Q are chosen. Without loss of generality we may assume that all H_2 queries are distinct (as the replies can be cached), and that any query involving $\mathsf{ID}A$ is issued (be it encryption, decryption or key generation) implies that $\mathcal{A}$ has already issued an H_2 query for A.

As before we may make assumptions about the behaviour of $\mathcal{A}$ when issuing decryption queries. Firstly, we may assume $\mathcal{A}$ does not issue a decryption query for a ciphertext that was the result of a previous encryption query, and similarly, it never issues a decryption query for a ciphertext when it has previously issued a key generation query on the sender or receiver ID.

An H_2 query on an $\mathsf{ID}A$ is handled thus:

1. If it is the ith query, respond with g^b. We call A a guessed ID.
2. If it is the jth query, respond with g^c. We call A a guessed ID.
3. Otherwise, choose a random $d \stackrel{\mathrm{R}}{\leftarrow} \mathbb{F}_q$, insert the tuple $\langle A, d\rangle$ into L_2 and return g^d.

For every H_1 query on a tuple $\langle r, h\rangle \in \mathbb{F}_q \times G_2$, a random $R \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ is output, and the tuple $\langle r, h, R\rangle$ is inserted into the list L_1.

Next, $\mathcal{A}$'s queries of step 2 are handled:

1. **Encryption**: suppose $\mathcal{A}$ issues an encryption query for a plaintext M between ID's A and B. If A and B are the guessed ID's, then B picks a random $r \stackrel{\mathrm{R}}{\leftarrow} \mathbb{F}_q$, a random $K \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and a random $R \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$. Then $\mathcal{B}$ outputs the ciphertext $C := \langle r, R, E_K(M)\rangle$.

   Otherwise without loss of generality assume A is not a guessed ID. Let d_B be the response to the H_2 query that was issued for B. In this case the list L_2 must contain the entry $\langle A, d\rangle$ for some $d \in \mathbb{F}_q$. Then A's private key is g^{ad}, and the ciphertext is computed as described by the $\mathsf{Authenticated\text{-}Encrypt}$algorithm (i.e. $s := e(d_B, g^{ad})$ is computed and so on). $\mathcal{A}$ is given the ciphertext.
2. **Decryption**: whenever $\mathcal{A}$ issues a decryption query, it is notified that the given ciphertext is invalid. Since we assume our scheme is ciphertext unforgeable, $\mathcal{A}$ cannot distinguish between this simulation of a decryption oracle and a real one (with overwhelming probability).
3. **Key generation**: suppose $\mathcal{A}$ issues a key generation query for an $\mathsf{ID}A$. If A is a guessed IDthen $\mathcal{B}$ fails. Otherwise the list L_2 must contain the entry $\langle A, d\rangle$ for some d and $\mathcal{B}$ outputs g^{ad}.

Eventually $\mathcal{A}$ outputs any two plaintexts $M_0, M_1 \in \mathcal{M}$ and any two ID's A, B on which it wishes to be challenged on. If these are not the guessed ID's then $\mathcal{B}$ fails. Otherwise $\mathcal{B}$ picks a random $U \stackrel{\mathrm{R}}{\leftarrow} \mathbb{F}_q$, a random $V \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$, and a random $\sigma \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$. It then computes $W := E_\sigma(M)$ for any (not necessarily random) message M, and responds with the challenge $C := \langle U, V, W\rangle$.

The next round of queries is handled as before.

Finally $\mathcal{A}$ outputs its guess. This is ignored.

The probability $\mathcal{A}$ does not ask for the keys of one of the guessed ID's is again at least $1/\binom{Q}{2}$.

The probability $\mathcal{A}$'s challenge ID's are the guessed ID's is also least $1/\binom{Q}{2}$.

Now if $\mathcal{A}$ has never queried $H_1(U, s)$ for $s = e(g, g)^{abc}$, since we are modeling H_1 and H_4 as random oracles, its view is independent of M, and so in this case $\mathcal{A}$ is unable to tell that that it is in a simulation, and has no advantage. Hence the probability that $\mathcal{A}$ queries $H_1(U, s)$ is at least ε.

If $\mathcal{A}$ has queried $H_1(U, s)$ then it may be able to distinguish the simulation from real life (it can tell that ciphertexts generated by the simulation are invalid), but s will be recorded on L_1. Then $\mathcal{B}$ wins provided it guesses the correct element of L_1 to output. The size of this list is bounded by Q_1.

4.5 Non-identity-based Authenticated Encryption

It is also not difficult to attain authenticated communication using standard cryptographic constructions. For example, if one uses a SKIP-like system [14], then Alice's public and private keys are g^x, x, say, and Bob's are g^y, y. Then they can use the quantity g^{xy} as a "shared secret", i.e. in the same manner we have used $s = e(H_2(A), d_B) = e(H_2(A), H_2(B))^a = e(d_A, H_2(B))$ to acheive authenticated encryption and decryption. (If we assume CDH is hard, no one but Alice and Bob can efficiently compute g^{xy}.)

4.6 Practical Considerations

The Fujisaki-Okamoto construction [8] used in the Boneh-Franklin system and our system facilitates proofs of security and allows for theoretically clean, self-contained, standalone systems. However, for practical purposes, it is more convenient to forgo the Fujisaki-Okamoto construction and instead achieve chosen ciphertext security by combining a slightly modified IBE system with semantically secure symmetric encryption schemes and message authentication schemes (that are secure against existential forgery) as described by Shoup [13, Theorem 1], Krawczyk [9, Theorem 1], and others for any public key system.

The Boneh-Franklin scheme is easily converted to a key encapsulation mechanism (see Shoup [13] for the definition). This (identity-based) key encapsulation mechanism can then be composed with encryption and MAC schemes to yield a chosen-ciphertext secure IBE system [13, 9]. Briefly, using our notation, the Boneh-Franklin key encapsulation mechanism is defined as follows: generating an encryption of a random key for an IDA consists of picking a random $r \stackrel{R}{\leftarrow} \mathbb{F}_q$ and outputing g^r. This g^r is the encryption of the random key $K = H(e(H_2(A), g^a)^r)$, where H is a hash function $H\colon G_2 \times \{0,1\}^n \to \mathbb{F}_q$. If H is modeled as a random oracle then it is readily shown that this is indeed a secure key encapsulation mechanism.

Similarly, for our scheme, the quantity we denote by s can be used as a shared secret to generate additional keys which are then used to encrypt and authenticate the message with off-the-shelf symmetric encryption and MAC schemes.

These modified schemes are often used instead of their original counterparts in the real world [15].

5 Conclusion

We have given formal definitons of security and integrity for identity-based encryption schemes. We have constructed an authenticated identity-based encryption scheme by extending the Boneh-Franklin

scheme, and shown that it is secure and authenticated using only the BDH assumption and the random oracle model.

If non-repudiation is desired, identity-based signature schemes can be used. It is worth noting that the Boneh-Franklin scheme can be extended to allow for identity-based signatures [4], that is, the same system parameters and public and private keys are used for signatures. However, this scheme is less efficient than authenticated encryption, as encryption and signing are separate operations; it is natural to ask if there is a way to perform identity-based signcryption (encryption with built-in signatures) just as efficiently as anonymous or authenticated encryption.

References

[1] G. Appenzeller and B. Lynn, "Stateless Network Layer Security", work in progress.

[2] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing", *Advances in Cryptology: CRYPTO 2001 (LNCS 2139)*, pp. 213–229, 2001.

[3] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing", *Cryptology ePrint Archive*, Report 2001/090, 2001. `http://eprint.iacr.org/2001/090/`

[4] Jae Choon Cha and Jung Hee Cheon, "Identity-based Signature from the Weil pairing", submitted for publication

[5] J. Coron, "On the exact security of Full-Domain-Hash", Proc. of Crypto 2000.

[6] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity", *J. Cryptology*, vol. 1, pp. 77–94, 1988.

[7] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", Proc. Crypto '86, pp. 186–194, 1986.

[8] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes", *Advances in Cryptology: CRYPTO '99 (LNCS 1666)*, pp. 537–554, 1999.

[9] H. Krawcyzk, "The Order of Encryption and Authentication for Protecting Communications", Proc. Crypto '01, 2001.

[10] R. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret", Proc. Asiacrypt '01, pp. 552–565, 2001.

[11] R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems Based on Pairing", SCIS2000

[12] A. Shamir, "Identity-based cryptosystems and signature schemes", *Advances in Cryptology: CRYPTO '84 (LNCS 196)*, pp. 47–53, 1985.

[13] V. Shoup, "Using Hash Function as a Hedge Against Chosen Ciphertext Attack", Proc. Eurocrypt 2000

[14] SKIP, "Simple Key management for Internet Protocols", `http://skip.incog.com/`

[15] B. Lynn, The Stanford IBE Library, `http://crypto.stanford.edu/ibe/`