1 Overview

1.1 Before
Deep into IBE & Sigs
- Different Assumptions
- Random Oracles

1.2 Now - Phase Shift
New Functionality
- Aggregate
- Proof of Storage
- Batching

2 BLS-'01

2.1 Algorithms
BLS Signature Scheme is based off of the Boneh-Franklin IBE Scheme. It uses the fact that the Computational Diffie-Hellman problem is hard to form a signature scheme. Let $G$ be a large prime order group with generator $g$. Let $H$ be a hash function that maps elements of $\{0,1\}^*$ to $G$.

- Setup($\gamma$):
  1. $G, H, g, g^y = Y, H: \{0,1\}^* \rightarrow G$
  2. $SK = y \in \mathbb{Z}_p$
- Sign($M, Sk$) $\rightarrow \sigma = H(M)^y$
- Verify($M, \sigma$) $\rightarrow e(\sigma, g) = e(H(M), Y)$

Setup takes in a security parameter and returns a secret key $SK$ equal to $y$, and a set of public parameters $PP$, which are $G$, $H$, $g$, and $Y$. Sign takes in a message $M$ and a secret key $SK$ and returns $\sigma$ which is the the hash function on $M$ raised to the $y$. Verify takes in a message $M$ and a signature $\sigma$ and returns true if the signature matches the Message and false otherwise.
2.2 Proof of Security

Proof.
We want to prove security based on the Computational Diffie-Hellman problem, which is
given a group \( G \), a generator \( g \), and 2 group elements \( A = g^a \) and \( B = g^b \) for some \( a, b \in \mathbb{Z}_p \),
compute \( g^{ab} \).
Assume attacker A on signature scheme makes \( Q \) queries. We can build a reduction \( B \) from
the signature scheme that breaks CDH.

1. Guess \( K \in [1, Q] \), the set of all integers from 1 to \( Q \), hope adversary forges on Kth
query, abort if not Kth query
2. \( PK = g, Y = g^a \)
Hash Table:

- \( K \)th query = \( g^b \)
- ith such that \( i \neq k \) - pick \( x_i \in \mathbb{Z}_p \) output \( g^{x_i} \)
Signature for \( M_i = Y^{x_i} = H(M)^y \) for all \( i \neq K \). If the attacker asks us for Kth signature,
abort because we guessed the wrong \( K \).
Attacker guesses \( \sigma^* \) for \( M_K \). If verify(\( M_K, \sigma^* \)) = true, then \( e(\sigma^* = g^{ab}, g) = e(H(M_k) = g^a, Y = g^b) \). That implies that \( \sigma^* = g^{ab} \). Thus we guess \( \sigma^* \) to try to break CDH.

\( \square \)

Side note about Proof:
We proved the signature scheme under CDH, which is a weaker assumption than DBDH,
which we used to prove Boneh-Franklin. Thus the signature scheme is stronger than the
IBE scheme.

3 Aggregate Signatures

3.1 Overview

Given a set of public keys, messages, and signatures \((PK_1, \sigma_1, M_1)\)...\((PK_n, \sigma_n, M_n)\), is there
a way to reduce the amount of signatures while still being able to verify,
\ie \( M_1, ..., M_n, \sigma_{agg} \)
When would this not be interesting?

- If \( \sigma_{agg} \) is concatenation or too big \( \rightarrow \) \( \sigma_{agg} \) same size as \( \sigma_{1...n} \)
- If \( M \)'s are too big, signature's size not important

When would this be interesting or useful?

- Petition signing \((M_1 = M_2 = ... = M_n, \sigma_1 \neq \sigma_2 \neq ... \neq \sigma_n)\)
- BGP routing
  - Trying to make it secure, but only have little space for signatures
  - Message is implicit, so can use aggregate signatures
3.2 Algorithms

We would still need to setup and sign individual messages, so Setup and Sign are the same as they are for normal signature schemes. However we need to somehow deal with aggregating signatures.

- Setup → PK, SK
- Sign(SK, M) → (σ, M)
- Aggregate(σ_agg (M_1, ..., M_k), (PK_1, ..., PK_k), σ, M_{k+1}, PK_k) → (σ^*_{agg}, M_{k+1}), (PK_{k+1}, ..., PK_{k+1})
- VerifyAgg(PK_1, ..., PK_n, M_1, ..., M_n, σ_{agg}) → {0,1}

Aggregate takes an aggregate signature σ_agg (anywhere between 1 and k signatures aggregated together) and a new signature σ, and returns a new aggregate signature σ^*_{agg}. VerifyAgg checks the aggregate signature against the group of messages, and returns true if they match, and false otherwise.

3.3 Security Definition

The attacker is successful iff the aggregate signature verifies and M^* is not queried, however M_1, ..., M^n could have been queried. The forgery could be aggregate and include some queried signatures, but σ^* must be used to compute σ_{agg}.

3.4 Aggregate BLS

The BLS Aggregate signature scheme is similar to the general BLS. We assume that we always use the same group G and the same hash function H. It uses the same setup and Sign algorithms, however we need to add an Aggregate algorithm and modify the Verify algorithm to allow aggregate signatures.

- Setup(γ):
  1. G, H, g, g^y = Y, H:{0,1}* → G
  2. SK = y ∈ Z_p
  3. PK = g^y
When setup is run multiple times we get $PK_1 = g^{y_1}, \ldots, PK_n = g^{y_n}$

- $\text{Sign}(M, Sk) \rightarrow \sigma = H(M)^y$

When setup is run multiple times we get $\sigma_1 = H(M)^{y_1}, \ldots, \sigma_n = H(M)^{y_n}$

- $\text{Aggregate}(\sigma_{agg} (M_1, \ldots, M_n), \sigma, M)$
  - $\sigma_{agg} = \sigma_{agg} \cdot \sigma_{n+1} = H(M_1)^{y_1} \cdot H(M_1)^{y_1} \cdot \ldots \cdot H(M_n)^{y_n+1} \in \mathbb{G}$

- $\text{VerifyAgg}$
  - first attempt: $e(g^{y_1}, g^{y_1} \ldots g^{y_1}, \prod H(M_i)) = ? e(\sigma_{agg}, g)$
  - second attempt: $\prod e(g^{y_i}, H(M_i)) = ? e(\sigma_{agg}, g)$

The first attempt at $\text{VerifyAgg}$ does not work because for each individual $H(M_i)$, we need to find that raised to the $y_i$ before the multiplication occurs. It does not suffice to simply multiply $g^{y_i}$ together, and multiply $H(M_i)$ together for all $i$ and then pair them. Thus we needed to change to the algorithm in the second attempt.

But is the above signature scheme actually secure?

An attack exists where attacker creates PK without knowing SK and uses it to attack.

$Y_B = g^{y_{Brent}}, Y_A = Y_B^{-1}$

$\sigma_{agg} = H(M^*)^{Y_B} \cdot H(M^*)^{-Y_B} = 1$

This is an easy problem to fix. If instead of sending just a message, we send $M^*: Y_B$, or the message concatenated with the public key, we avoid this problem.

The attack would turn into:

$Y_B = g^{y_{Brent}}, Y_A = Y_B^{-1}$

$\sigma_{agg} = H(M^* : Y_B)^{Y_B} \cdot H(M^* : Y_A)^{-Y_B} = 1$

Since $M^* : Y_B$ can’t be sent if PK = $Y_A$, the attacker must send a different message, making it secure.

Proof of Aggregate BLS is similar to general BLS