BLS Signature and GPSW Attribute-Based Encryption

2019-2020

Contents

1 Boneh-Lynn-Schucham signature

Definition 1. *A signature scheme is a tuple of 3 PPT algorithms (Setup,Sign,Verif) such that*

- $\mathrm{Setup}(1^\lambda)$*: outputs a pair of public/secret key* (pk, sk)
- $\mathrm{Sign}(sk, m)$*: outputs a signature* σ *of message* m*.*
- $\mathrm{Verif}(pk, m, \sigma)$*: ouptputs 0 or 1.*

and with the following properties:

- Correctness*:* $\forall m, \mathbb{P}_{pk,sk \xleftarrow{\$} \mathrm{Setup}(1^\lambda)}[\mathrm{Verif}(pk, m, \mathrm{Sign}(sk, m)) = 1] \geqslant 1 - \mathrm{negl}(\lambda)$
- Unforgeability*: Adversary* $\mathscr{A}$ *gets pk from Challenger. Adversary* $\mathscr{A}$ *can adaptively query a* Sign *oracle with messages* $m_1, \dots, m_q$ *of its choice to get* $\sigma_i = \mathrm{Sign}(sk, m_i)$*.*
  For all PPT adversary $\mathscr{A}$*, we want*

  $$\mathbb{P}[A^{Sign(sk,.)}(pk) = (m^*, \sigma^*) | Verif(pk, m^*, \sigma^*) = 1] = \mathrm{negl}(\lambda)$$

  such that $\forall i = 1, \dots, q,\ m^* \neq m_i$*.*

Remark. *If Sign is randomized we talk about* strong *unforgeability if it is hard to produce a new* σ *even for one of the querried messages* m_i*'s.*
But BLS is deterministic, so Unforgeability = Strong unforgeability.

1.1 Constrution of BLS signature

- Setup : $pk = g^s$, $sk = s$, $H : \{0,1\}^* \to G$.
- Sign(sk, m): Compute $h = H(m)$ and return $\sigma = h^s = H(m)^s$.
- Verif(pk, m, σ): Compute $h = H(m)$ and output $e(g, \sigma) == e(h, pk)$.

1.2 Unforgeability

Theorem 1. *BLS satisfies unforgeability under the CDH assumption in G.*

Reminder. *DBDH assumption implies CDH in G.*

Proof. Let $\mathscr{A}$ be an adversary against unforgeability, $\mathscr{B}$ against CDH in G. $\mathscr{B}$ gets (g, g^a, g^b) and wants to compute g^{ab}.
$\mathscr{B}$ sets $pk = g^a$ (so $s = a$ implicitly).
If $H(m^*) = g^b$ then success, since a valid signature for m^* is $H(m^*)^s = (g^b)^a = g^{ab}$.
The only thing that remains to be done is to make sure that $H(m^*) = g^b$.
Let Q_H denote the number of oracle access to H made by $\mathscr{A}$. Just embed g^b as the ouptut of the i-th query for $i \xleftarrow{\$} \{1, ..., Q_H\}$. For other random oracle querries, output g^t for $t \xleftarrow{\$} \mathbb{Z}_p$.
$\Rightarrow$ Given t, one can compute $\sigma = H(m)^s = (g^s)^t$. □

BLS has several nice features: for s_1, s_2 and $\sigma_1 = H(m)^{s_1}, \sigma_2 = H(m)^{s_2}$, one can check $e(g, \sigma_1, \sigma_2) = e(H(m), g^{s_1}.g^{s_2})$. $\Rightarrow$ Verify poly-many signatures at once: "Agregate signatures".
$|\sigma| = |\text{group element in } G| \to$ very small.

2 GPSW ABE Scheme

2.1 Attribute-based encryption

Definition 2. *An attribute-based encryption scheme is a tuple of 4 PPT algoritms* (Setup, KeyGen, Enc, Dec) *such that:*

- Setup(1^λ)*: outputs a pair of master public/secret key (mpk, msk).*
- KeyGen(msk, P)*: on input msk and a predicate P, outputs sk_P.*
- Enc(mpk, γ, m)*: on input mpk, a set of attributes γ and message m, outputs a ciphertext ct.*
- Dec(sk_P, ct)*: outputs a message or $\perp$.*

γ is a subset of $\mathscr{U}$, the set of attributes. $\mathscr{U} = \{A_1, ..., A_n\}$
P is a predicate over $\mathscr{U}$, i.e:$P(\gamma) = 1 \Leftrightarrow \gamma \in S_P \subseteq 2^{\mathscr{U}}$. Equivalently: $P : \{0,1\}^n \to \{0,1\}$.

And such that the following properties hold:

- *Correctness:* $\mathbb{P}[\text{Dec}(sk_P, (\text{Enc}(mpk, \gamma, m))) = m] \geqslant 1 - \text{negl}(\lambda)$ *if* $P(\gamma) = 1$.
- `IND-CPA` *security:*

$$
\begin{array}{llcc}
 & \mathscr{A} & & C \\
\textit{Init} & & & (mpk, msk) \xleftarrow{\$} \mathrm{Setup}(1^\lambda) \\
 & & \xleftarrow{mpk} & \\
\textit{Phase 1} & & \xrightarrow{P_i} & \circlearrowleft \textit{ repeat } q \textit{ times} \\
 & & \xleftarrow{sk_{P_i}} & \\
\textit{Challenge} & \forall i, P_i(\gamma) \neq 1 & \xrightarrow{\gamma, m_0, m_1} & b \xleftarrow{\$} \{0,1\} \\
 & & \xleftarrow{ct_b} & ct_b = \mathrm{Enc}(mpk, \gamma, m_b) \\
\textit{Phase 2} & \forall i, P'_i(\gamma) \neq 1 & \xrightarrow{P'_i} & \circlearrowleft \textit{ repeat } q' \textit{ times} \\
 & & \xleftarrow{sk_{P'_i}} & \\
\textit{Finalize} & \downarrow b' & &
\end{array}
$$

- *Init: Challenger picks* $(mpk, msk) \xleftarrow{\$} \mathrm{Setup}(1^\lambda)$ *and sends* mpk *to* $\mathscr{A}$*.*
- *Phase 1: The adversary ask for keys for predicates* $P_1, ..., P_q$ *of its choice and gets* $sk_{P_1}, ..., sk_{P_q}$*.*
- *Challenge:* $\mathscr{A}$ *sends a set of target attributes* $\gamma^* \subseteq \mathscr{U}$ *and* m_0, m_1 *to* C *and gets* $\mathrm{Enc}(mpk, \gamma^*, m_b)$ *for* $b \xleftarrow{\$} \{0,1\}$*.* C *rejects if for some* $i \in \{1, ..., q\}, P_i(\gamma^*) = 1$*.*
- *Phase 2: Same as Phase 1, with* $P_i(\gamma^*) \neq 1$ *for each new query.*
- *Finalize:* $\mathscr{A}$ *outputs a guess* b' *and wins if* $b' = b$*.*

Remark. *We define selective security (`sel-IND-CPA`) in a analogous way than standard `IND-CPA` security except that in the selective setting, the adversary picks the target attribute set* γ^* *BEFORE seeing the public key.*

Exemple 1. $\mathscr{U} = \{$*ENS, M2, Crypto, Lyon, Student*$\}$.
$P = (ENS \wedge Lyon) \vee (M2 \wedge Student \wedge Crypto)$.
Here, given sk_P*, we can decrypt a message for* $\gamma_1 =$ *(ENS, Lyon) but not for* $\gamma_2 =$ *(Lyon, M2).*

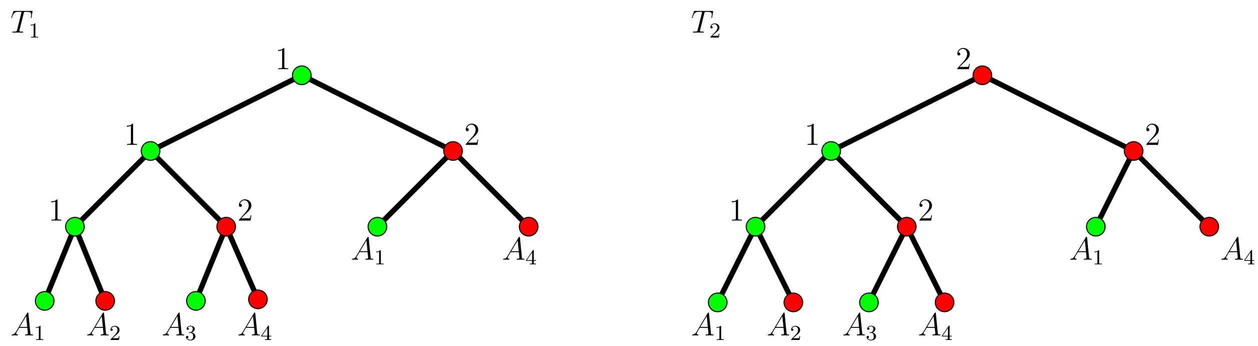
Remark. *IBE = ABE for* $\mathscr{U}$ *= set of ID's and* $P = \{P_{ID}(ID') = (ID == ID') | ID \in \mathscr{U}\}$*.*

2.2 The GPSW'06 construction

$\mathscr{U} = \{A_1, ..., A_n\}$ and P is an access tree.

Definition 3. *An access tree* τ *with variables* $\{A_1, ..., A_n\}$ *is a tree with every internal node* x *being labeled by some threshold* $0 \leqslant k_x \leqslant d_x$ *(*d_x*: degree of the node* x*) and leaves are labeled by a variable* A_i*.*
A leaf with label A_i *evaluates to* 1 *on a set of attributes* $\gamma \subseteq \{A_1, \ldots, A_n\}$ *if* $A_i \in \gamma$*. Now denote* T_x *the subtree with root* x*, then* T *with root* r *evaluates to 1 on a set of attributes* $\gamma \subseteq \{A_1, \ldots, A_n\}$ *if there exists (at least)* k_r *children of* r *such that the subtree rooted in each of these children evaluate to 1.*

Exemple 2. *Let* $\gamma = \{A_1, A_3\}$*. Then* $T_1(\gamma) = 1$*, but* $T_2(\gamma) = 0$*.*



Remark. *If $k_x = d_x$ then x compute an AND, if $k_x = 1$ then x compute an OR.*

The GPSW Contruction for access trees:

- Setup(1^λ): $s \xleftarrow{\$} \mathbb{Z}_p$, $t_1, ..., t_n \xleftarrow{\$} \mathbb{Z}_p$.
  Outputs $mpk = \{e(g,g)^s, g^{t_1}, ..., g^{t_n}\}$ and $msk = \{s, t_1, ..., t_n\}$.

- Enc(mpk, γ, m): $r \xleftarrow{\$} \mathbb{Z}_p$.
  Outputs $ct = (m.e(g,g)^{sr}, \{g^{t_i r}\}_{A_i \in \gamma})$.

- KeyGen(msk, T): Recursively define polynomials of every node of T from the root r to the leaves as follows:

  - q_r is a random polynomial of degree $k_r - 1$ such that $q_r(0) = s$.
  - For every node x, define q_x as a degree $(k_x - 1)$ random polynomial such that $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ where index(x) is the index of x as a children of parent(x) (i.e., index(x) is a unique number between 1 and $d_{\text{parent}(x)}$ associated to x.

  sk_P is defined by: $\{g^{\frac{q_x(0)}{t_i}} | x \text{ is a leaf with attribute } A_i\}$.

- Dec(sk_T, ct): Lagrange interpolation from leaves to root starting with $e(g^{\frac{q_x(0)}{t_i}}, g^{t_i.r})$.
  Thanks to the linearity of Lagrange interpolation, given $d+1$ group elements of the form $g^{p(i_1)}, \ldots, g^{p(i_{d+1})}$ with p a degree-d polynomials and $i_1 \neq \cdots \neq i_{d+1}$, one can recover $g^{p(0)}$.

  If $T(\gamma) = 1$, one can then recover $e(g,g)^{sr}$ by interpolating from leaves to root starting with $e(g,g)^{q_x(0)r}$ for every leaf $x \in T$ and going from leaves to the root of T.

Theorem 2. *The GPSW'06 is* `sel-IND-CPA` *secure under DBDH assumption.*

Proof. Similar to BF'01:

- Game 0: $ct = \text{Enc}(mpk, \gamma, m_0) = (m_0.e(g,g)^{sr}, \{g^{t_i.r}\}_i)$

- Hyb 1: $ct = (z, \{g^{t_i.r}\}_i)$ with $z \xleftarrow{\$} G_T$

- Game 1: $ct = \text{Enc}(mpk, \gamma, m_1) = (m_1.e(g,g)^{sr}, \{g^{t_i.r}\}_i)$

$\mathscr{A}$ declares the target set of attributes γ^*. $\mathscr{B}$ gets (g^a, g^b, g^c, z) with $z = e(g,g)^{abc}$ or $z \xleftarrow{\$} G_t$. The idea is to set $sr = abc$, so set $ab = s$ and $r = c$, where s is the master secret key and r denote the randomness used for the challenge ciphertext. Doing so, $\mathscr{B}$ sets $mpk = \{e(g,g)^{ab}, g^{t_i}\}$ for some $t_i \in \mathbb{Z}_p$. We detail later how the t_i's are picked (this part will depend on γ^*, which is why this proof only gives `sel-IND-CPA` security).

The main technicality in the proof is to provide $\mathscr{B}$ with a way to generate keys for $\mathscr{A}$ since $\mathscr{B}$ does not know ab nor g^{ab} but only g^a and g^b.

Consider a query T made to KeyGen by $\mathscr{A}$, so that $T(\gamma^*) = 0$. Then $\mathscr{B}$ generates a key for T as follows. B runs a similar process than the one used in the KeyGen algorithm. Starting at the root r associated with degree k_r of T, it implicitly defines degree $k_r - 1$ polynomial q_r such that $g^{q_r(0)} = g^a$. Since $T(\gamma^*) = 0$, there are at most $k_r - 1$ children of r $x_1, \ldots, x_{k_r-1}$ such that subtrees they are the roots of evaluate to 1 in γ^*. Then, $\mathscr{B}$ picks up to $k_r - 1$ random points $y_1, \ldots, y_{k_r-1}$ in $\mathbb{Z}_p$ and sets $q_r(\text{index}(x_i)) = y_i$. It then defines recursively polynomials q_x for every internal node x of T such that $g^{q_x(0)} = g^{q_{\text{parent}(x)}(\text{index}(x))}$. If the subtree rooted in x is not satisfied γ^*, then one might know only $g^{q_{\text{parent}(x)}(\text{index}(x))}$ but not $q_{\text{parent}(x)}(\text{index}(x))$, while if it is satisfied by γ^*, we are guaranteed to know it.

As we started with $q_r(0) = a$, this gives a key for $msk = a$ but not for $msk = ab$ as we wish. Yet, a valid key for $msk = ab$ is then the set of all $((g^{q_x(0)/t_i})^b)_i$ such that x is a leaf (associated with A_i).

There is one issue: if the leaf is not satisfied, it is possible that we only know $g^{q_x(0)}$ and not $q_x(0)$ in clear. This is an issue as we also do not know b but only g^b. The trick is then to have defined t_i as $t_i = bt'_i$, with $t'_i \xleftarrow{\$} \mathbb{Z}_p$ if $A_i \notin \gamma^*$ such that $(g^{q_x(0)/t_i})^b = g^{q_x(0)/t'_i}$ and then one can still compute the key without knowing b nor $q_x(0)$. If $A_i \in \gamma^*$, this is not an issue as we know $q_x(0)$ and can compute the corresponding key component from g^b. Yet, there is another issue if we let $t_i = bt'_i$ as well for $A_i \in \gamma^*$.

Indeed, to generate the challenge ciphertext, we need to output the $g^{t_i c}$ for all t_i's such that $A_i \in \gamma^*$. As we do not know g^{bc} but only g^b, g^c, we cannot generate these if t_i is also defined as $t_i = bt'_i$ for $A_i \in \gamma^*$, but we can if we choose $t_i \xleftarrow{\$} \mathbb{Z}_p$.

Therefore, $\mathscr{B}$ needs to know γ^* before generating the public key in order to correctly generate the g^{t_i}'s as either $(g^b)^{t'_i}$ if $A_i \notin \gamma^*$ or simply as g^{t_i} with chosen $t_i, t'_i \xleftarrow{\$} \mathbb{Z}_p$ otherwise.

It is now easy to conclude the proof. □

Remark. *While the above proof only achieves `sel-IND-CPA` security, note that the challenger can guess the target set of attributes γ^* with probability $1/2^n$. This artificial trick allows to go from selective to adaptive security by guessing the target challenge. It is often referred to as* complexity leveraging. *This does not give a stronger statement as there is a exponential loss in the reduction due to this guess, but the take-away is that proving selective security still provides reasonable security guarantees against adaptive adversaries if we increase sufficiently the parameters. Specifically, since n is independent of the security parameter, one can pick the groups such that DBDH is hard even for adversaries that run in time $2^n \cdot poly$.*