



THE POWER OF PAIRINGS TOWARDS STANDARD MODEL SECURITY

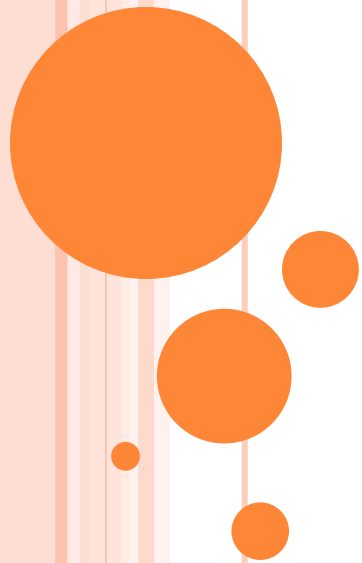
**Pairings, IBE, IND-CCA-secure encryption,
authentication**

FROM PREVIOUS LECTURE

- Public-key Crypto
 - Alternative to symmetric key primitives
 - Do not require sharing keys, but they require a PKI
- PKE
 - Comes in 2 flavours: IND-CPA and IND-CCA
 - Saw 1 construction based on DDH that is IND-CPA
 - Malleability implies no IND-CCA
- Signature Schemes
 - Security: EUF-CMA
 - RSA signatures are not EUF-CMA
 - But we could use FDH in the random oracle model



PART I PAIRINGS



PAIRINGS IN GENERAL

➤ Setting :

- 2 additive groups G_1, G_2 , multiplicative group G_T
- All three groups of prime order q
- We can write $G_1 = \langle P, \dots, qP \rangle$ and $G_2 = \langle Q, \dots, qQ \rangle$

➤ Imagine a mapping $e: G_1 \times G_2 \rightarrow G_T$ such that:

- Bilinear: for all $a, b \in \{1, \dots, q - 1\}$ it holds that:

$$e(aP, bQ) = e(P, Q)^{ab}$$

- Non-degenerate: $e(P, Q) \neq 1$
- Efficiently computable



PAIRINGS IN CRYPTOGRAPHY

- Usually computed on elliptic curves
- There are different types, depending on how the pairing is constructed
- Security depends on type and on something called “embedding degree”
- Mostly defined with elements from additive subgroups (rather than multiplicative ones), but we will keep the multiplicative notation
- We will not cover specifics in this course
 - If you’re interested, you could read:

Lawrence C. Washington:

‘Elliptic curves: Number theory and cryptography’



DDH AND PAIRINGS

- Consider multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order q , and a pairing $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}^T$ on this group
 - Given (g, g^a, g^b, g^c) DDH problem requires to decide whether $g^c = g^{ab}$ or g^c just random element
 - Bilinearity: $e(g^a, g^b) = e(g, g)^{ab} = e(g, g^{ab})$
 - DDH adversary tests whether $e(g^a, g^b) = e(g, g^c)$
 - If so, then guess that $g^c = g^{ab}$
 - Else, output that g^c is random
- Conclusion: DDH is easy to solve in groups that admit pairings



HARD PROBLEMS WITH PAIRINGS

- Setup: multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order q , given a bilinear mapping $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}^T$
- Computational Bilinear DH problem:
 - Given (g, g^a, g^b, g^c) , compute g^{abc}
- Decisional Bilinear DH problem
 - Given (g, g^a, g^b, g^c, g^z) , decide whether $g^z = g^{abc}$
- CDH and DLog:
 - We think these are still hard despite pairings



WHY WE USE PAIRINGS

Alice

Bob

Choose
 $a \in_R \{0, \dots, q - 1\}$

Choose
 $b \in_R \{0, \dots, q - 1\}$

$A = aP$

$B = bP$

Compute
 $K = bA$

Compute
 $K = aB$

Same K :
 $bA = baP = abP = aB$

$a \in_R \{0, \dots, q - 1\}$
 $A_1 = aP; A_2 = aQ$

Alice



A_1, A_2

Charlie

$c \in_R \{0, \dots, q - 1\}$
 $C_1 = cP; C_2 = cQ$



C_1, C_2

B_1, B_2



Bob

$b \in_R \{0, \dots, q - 1\}$
 $C_1 = bP; C_2 = bQ$



THREE-PARTITE KEY EXCHANGE

Alice

Bob

Choose
 $a \in_R \{0, \dots, q-1\}$

Choose
 $b \in_R \{0, \dots, q-1\}$

$$\longleftarrow A = aP$$

$$\longrightarrow B = bP$$

Compute
 $K = bA$

Compute
 $K = aB$

Same K :
 $bA = baP = abP = aB$

$$K = e(B_1, C_2)^a = e(bP, cQ)^a = e(P, Q)^{abc}$$

Alice

A_1, A_2

C_1, C_2

Charlie

$$K = e(A_1, B_2)^c = e(aP, bQ)^c = e(P, Q)^{abc}$$

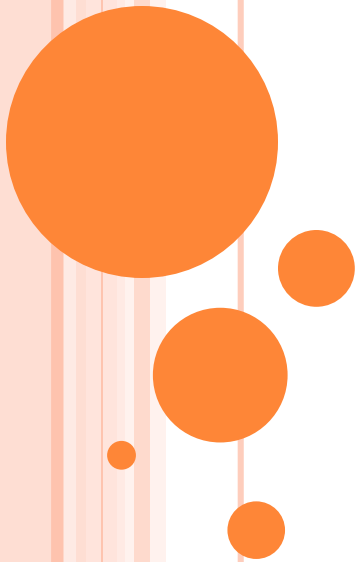
B_1, B_2

Bob

$$K = e(C_1, A_2)^b = e(P, Q)^{abc}$$



PART II
IDENTITY-BASED ENCRYPTION



PKE AND IBE

➤ PKE:

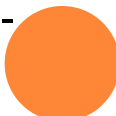
- Alice has a private key for decryption
- Bob (and everyone else) has a public key for encryption to Alice
- Problem of certification: whose key is that?

➤ IBE:

- Bob has (a function of) Alice's identity (name, email address, social security number) as a PK
- Alice can derive a secret key from that
- Bob encrypts with Alice's identity, so only she can decrypt



IBE SYNTAX

- Tuple of algorithms (Setup, KGen, Enc, Dec) with:
 - $\text{Setup}(1^\lambda)$: on input the security parameter, this algorithm outputs $(MSK, PPar)$, a master secret key and some global parameters
 - $\text{KGen}(MSK, ID)$: on input the master secret key and the identity, this algorithm outputs an identity-specific secret key sk_{ID}
 - $\text{Enc}(ID; M)$: on input an identity and a message, output a ciphertext c
 - $\text{Dec}(sk_{ID}, c)$: on input the identity-specific sk_{ID} and a ciphertext, output plaintext \hat{m} or symbol \perp
- 

IBE SETUP

- Why do we need a setup algorithm for IBE and not for regular PKE?



IBE SETUP

- Why do we need a setup algorithm for IBE and not for regular PKE?
- Not because we need MSK to generate our secret keys with
- After all, each user could just generate sk_{ID} as we do in regular PKE, right?



IBE SETUP

- Why do we need a setup algorithm for IBE and not for regular PKE?
- Not because we need MSK to generate our secret keys with
- After all, each user could just generate sk_{ID} as we do in regular PKE, right?
- Wrong!
- We need to ensure that the parameters are chosen well, so that there's no clash for sk_{ID} !



PAIRING BASED IBE

- Designed by Boneh and Franklin in 2001
- Ingredients:
 - Identity space \mathbb{ID}
 - A hash function (will see it later)
 - A bilinear mapping
- Setup outputs:
 - A couple of groups \mathbb{G}, \mathbb{G}^T of prime order q
 - A secret value $y \in \{1, 2, \dots, q - 1\}$
 - A generator g for \mathbb{G} , and the value g^y
 - A hash function $H: \mathbb{ID} \rightarrow \mathbb{G}$
 - Set $MSK = y$; $PPar = (\mathbb{G}, \mathbb{G}^T, g, g^y, H)$



BONEH-FRANKLIN IBE

- $MSK = y$; $PPar = (\mathbb{G}, \mathbb{G}^T, g, g^y, H)$
- ID-specific secret key generation:
 - Takes input y, ID
 - Output $sk_{ID} = H(ID)^y \in \mathbb{G}$
- Encryption:
 - Takes input m, ID
 - Choose random $r \in \{1, \dots, q - 1\}$, compute g^r
 - Output: $c = (g^r, m \cdot e(H(ID), g^y)^r)$
- Decryption:
 - Takes input $c = (c_1, c_2), sk_{ID}$
 - Compute: $c_2 / e(H(ID)^y, g^{c_1}) = \hat{m}$



SECURITY OF BONEH-FRANKLIN

➤ Theorem:

- BF is IND-CPA in the random oracle model if the Decisional Bilinear DH problem is hard in \mathbb{G}

➤ Translation:

- In the random oracle model
- If there exists an adversary that wins the IND-CPA game against the BF scheme with probability $\frac{1}{2} + p_A$
- Then there exists an adversary B that can solve the DBDH problem in \mathbb{G} with probability $\frac{1}{2} + \frac{1}{2q_H} p_A$,



IND-CPA FOR IBE

- IND-CPA: eavesdropper can't tell even 1 bit of p-text

$$(MSK, PPar) \leftarrow \text{Setup}(1^\lambda)$$

$$b \leftarrow_{\$} \{0,1\}$$

$$(m_0, m_1, ID) \leftarrow \mathcal{A}^{KGen(\cdot), H(\cdot)}(PPar, 1^\lambda)$$

$$c \leftarrow \text{Enc}(ID; m_b)$$

$$d \leftarrow \mathcal{A}^{KGen(\cdot)}(c, PPar, 1^\lambda)$$

\mathcal{A} wins iff. $d = b$ and $KGen(ID)$ never queried

Parameter: q_H RO queries

- Intuition: we will need the ROM in order to make sure that the small entropy from identifiers translates to a LOT of entropy for the secret keys



PROOF OF IND-CPA OF BF

➤ Proof:

- B's goal is to distinguish between $(g, g^a, g^b, g^c, g^{abc})$ and (g, g^a, g^b, g^c, g^z)
- B's strategy will be to inject the challenge into a single identity ID ; then B will hope that A will output THAT identity for the challenge
- Constructing B:
 - Receives (g, g^a, g^b, g^c, g^z) with z random or $z = abc$
 - Begin by running Setup, need to output $Ppar$ to A
 - Insert $g^y = g^a$, output $(\mathbb{G}, \mathbb{G}^T, g, g^a, H)$ to A
 - A can now make $KGen$ and H queries
 - The former outputs secret keys, but not for the challenge ID
 - The latter allows to just hash identities (in the ROM)



PROOF OF IND-CPA OF BF

➤ Proof (continued):

▪ Constructing B:

- Receives (g, g^a, g^b, g^c, g^z) with z random or $z = abc$
- Begin by running Setup, need to output $Ppar$ to A
 - Insert $g^y = g^a$, output $(\mathbb{G}, \mathbb{G}^T, g, g^a, H)$ to A
- A can now make $KGen$ and H queries
 - B: guesses a random index: $i \in \{1, \dots, q_H\}$
 - Answer to H queries (programming RO):
 - On j -th query, $j \neq i$, pick random r_j , output $H(x) = g^{r_j}$
 - On i -th query, insert $H(x) = g^b$
 - Answer to $KGen$ queries:
 - B knows $DLog$ of all $H(x)$, except for the i -th query
 - But A can't query the SK for that if it's his challenge



PROOF OF IND-CPA OF BF

➤ Proof (continued):

▪ Constructing B:

- Receives (g, g^a, g^b, g^c, g^z) with z random or $z = abc$
- Running Setup, output $(\mathbb{G}, \mathbb{G}^T, g, g^a, H)$ to A
- Answer to queries:
 - B: guesses a random index: $i \in \{1, \dots, q_H\}$
 - Answer to H queries (programming RO):
 - On j -th query, $j \neq i$, pick random r_j , output $H(x) = g^{r_j}$
 - On i -th query, insert $H(x) = g^b$
 - Answer to KGen queries:
 - On j -th query, output $SK_{ID} = (g^a)^{r_j} = H(x)^a$
 - On i -th query, abort
- A's challenge: A outputs (m_0, m_1, ID)
 - If ID was not i -th query, abort
 - Else: choose random b^* , output $(g^c, M_{b^*} \cdot e(g, g^z))$



PROOF OF IND-CPA OF BF

➤ Proof (continued):

- Receives (g, g^a, g^b, g^c, g^z) with z random or $z = abc$
- Running Setup, output $(\mathbb{G}, \mathbb{G}^T, g, g^a, H)$ to A
- Answer to queries:
 - B: guesses a random index: $i \in \{1, \dots, q_H\}$
 - Answer to H queries (programming RO):
 - On j -th query, $j \neq i$, pick random r_j , output $H(x) = g^{r_j}$
 - On i -th query, insert $H(x) = g^b$
 - Answer to KGen queries:
 - On j -th query: $SK_{ID} = (g^a)^{r_j} = H(x)^a$; if $j = i$, abort
- A's challenge: A outputs (m_0, m_1, ID)
 - If ID was not i -th query, abort and guess if $z = abc$ or not
 - Else: choose random b^* , output $(g^c, M_{b^*} \cdot e(g, g^z))$
- A's response: guess d^* of b^*
 - B guesses $z = abc$ iff. $d^* = b^*$



PROOF OF IND-CPA OF BF

➤ Proof (cont):

▪ Analysis:

- B chooses the wrong i implies B had to guess (B wins w.p. $\frac{1}{2}$)

Happens w.p. $1 - \frac{1}{q_H}$

- B chooses the right i implies:

If $z = abc$ simulation of game is perfect; A wins w.p. $\frac{1}{2} + p_A$

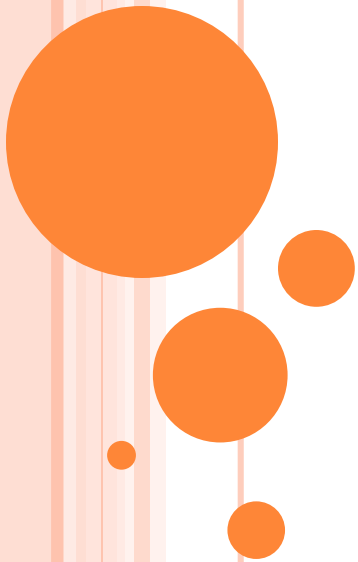
If z is random, c is statistically independent from m_0, m_1

A wins w.p. $\frac{1}{2}$

- B wins w.p.: $\frac{1}{q_H} \mathbb{P}[B \text{ wins} \mid B \text{ guesses right}] + \left(1 - \frac{1}{q_H}\right) \cdot \frac{1}{2} =$

$$\frac{1}{q_H} \left(\frac{1}{2} \left(\frac{1}{2} + p_A \right) + \frac{1}{2} \cdot \frac{1}{2} \right) + \left(1 - \frac{1}{q_H} \right) \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2q_H} p_A$$

PART II
THE USES OF IBE



FUJISAKI-OKAMOTO

- Designed a “compiler”:
 - Input: a PKE scheme that's IND-CPA secure
 - Output: a PKE scheme that's IND-CCA secure
- Boneh and Franklin used it on their IND-CPA scheme, and obtained an IND-CCA one
- We won't look at the generic compiler, but let's see the IND-CCA version of BF!
- For interested readers, see:

Fujisaki, Okamoto “Secure integration of asymmetric and symmetric encryption schemes”, Crypto 99



CCA-SECURE IBE

➤ Setup outputs:

- A couple of groups \mathbb{G}, \mathbb{G}^T of prime order q
- A secret value $y \in \{1, 2, \dots, q - 1\}$
- A generator g for \mathbb{G} , and the value g^y
- Hash functions: $H: \text{ID} \rightarrow \mathbb{G}$, $F: \{0,1\}^q \times \{0,1\}^q \rightarrow \mathbb{Z}_q^*$,
 $G: \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$
- Set $MSK = y$; $PPar = (\mathbb{G}, \mathbb{G}^T, g, g^y, F, G, H)$

➤ ID-specific secret key generation:

- Takes input y, ID
- Output $sk_{ID} = H(ID)^y \in \mathbb{G}$



IND-CCA VERSION OF BF

- Setup: $MSK = y$; $PPar = (\mathbb{G}, \mathbb{G}^T, g, g^y, F, G, H)$
- Key generation: $sk_{ID} = H(ID)^y \in \mathbb{G}$
- **Encryption:**
 - Takes input m, ID
 - Choose random $s \in \{0,1\}^q$, compute $r = F(s, m)$
 - Output: $c = (g^r, s \cdot e(H(ID), g^y)^r, m \cdot G(s))$
- **Decryption:**
 - Takes input $c = (c_1, c_2, c_3), sk_{ID} = H(ID)^y$
 - Compute: $c_2 / e(H(ID)^y, g^s) = \hat{s}$
 - Finally get $\hat{m} = c_3 / G(\hat{s})$



SECURITY STATEMENT

➤ Theorem:

- In the Random Oracle Model (F, G, H all ROs)
- If the DBDH assumption holds in group \mathbb{G} , then the modified Boneh-Franklin scheme is IND-CCA secure

- We will not prove this here
- Intuition: c_2 hides s like it hid m before, and we use s to hide m in c_3 . We use F to cryptographically bind r to s , but since F is a random oracle any change in s creates a random F output.



SIGNATURES IN THE STANDARD MODEL

- So far we've seen:
 - IND-CPA-secure encryption in the standard model (no ROs required) – ElGamal
 - IND-CPA-secure IBE in the ROM – Boneh-Franklin
 - IND-CCA-secure IBE in the ROM – BF + FO
 - EUF-CMA signatures in the ROM using Full-domain hashing (FDH)
- Let's see now:
 - (strongly) EUF-CMA signatures without random oracles, using pairings



STRONG UNFORGEABILITY

- EUF-CMA: adversary can't forge fresh signature

$$(sk, pk) \leftarrow \text{KGen}(1^\lambda)$$

$$(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}(\cdot)}(pk, 1^\lambda)$$

Store list $\mathcal{Q} = \{(m_1, \sigma_1), \dots, (m_k, \sigma_k)\}$ of queries to Sign

\mathcal{A} wins iff. $(m, \sigma) \notin \mathcal{Q}$ and $\text{Vf}(pk; m, \sigma) = 1$

- sEUF-CMA: adversary can't forge fresh signature

$$(sk, pk) \leftarrow \text{KGen}(1^\lambda)$$

$$(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}(\cdot)}(pk, 1^\lambda)$$

Store list $\mathcal{Q} = \{(m_1, \sigma_1), \dots, (m_k, \sigma_k)\}$ of queries to Sign

\mathcal{A} wins iff. $(m, \sigma) \notin \mathcal{Q}$ and $\text{Vf}(pk; m, \sigma) = 1$



STRONG UNFORGEABILITY: BSW

➤ Boneh, Shen, Waters

➤ Ingredients:

- Group \mathbb{G} of prime order q such that $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}^T$, with $\mathbb{G} = \langle g \rangle$
- Hash function $H: \{0,1\}^* \rightarrow \{0,1\}^n$ such that $q > 2^n$

➤ Key generation *KGen*:

- Choose secret $y \in \{1, \dots, q - 1\}$, compute g^y
- Choose public $g^*, h \in \mathbb{G}$, and random $u^*, u_1, \dots, u_n \in \mathbb{G}$
- Set $U = \{u_1, \dots, u_n\}$ and pick H
- Output: $PK = (g, g^a, g^*, h, u^*, U, H)$ and $SK = (g^*)^y$



STRONG UNFORGEABILITY: BSW

- *KGen* outputs $PK = (g, g^y, g^*, h, u^*, U, H)$ and $SK = (g^*)^y$
- Signing message m :
 - Pick random $r, s \in \mathbb{Z}_q$; Set $\sigma_2 = g^r \in \mathbb{G}$
 - Set $t \leftarrow H(m \parallel \sigma_2) \in \{0,1\}^n$; interpret t as element of \mathbb{Z}_q
 - Do $v \leftarrow H(g^t h^s) \in \{0,1\}^n$; write $v = v_1 \dots v_n$, with $v_i \in \{0,1\}$
 - Compute: $\sigma_1 = (g^*)^y (u^* \prod_{i=1}^n u_i^{v_i})^r$, output (σ_1, σ_2, s)
- Verification of signature (σ_1, σ_2, s) for message m :
 - Compute $\hat{t} = H(m \parallel \sigma_2)$, encode it as element of \mathbb{Z}_q
 - Do $\hat{v} \leftarrow H(g^{\hat{t}} h^s) \in \{0,1\}^n$; write $v = v_1 \dots v_n$, with $v_i \in \{0,1\}$
 - Verify: $e(\sigma_1, g) = e(\sigma_2, u^* \prod_{i=1}^n u_i^{v_i}) \cdot e(g^y, g^*)$



STRONG UNFORGEABILITY OF BSW

➤ Theorem:


- Given the hash function H is collision resistant
- Given the CDH is hard to solve in group \mathbb{G}
- Then the BSW scheme is strongly EUF-CMA

➤ Proof:

- Goal of sEUF-CMA attacker: output tuple $(m^*, (\sigma_1, \sigma_2, s))$ such that $(m^*, (\sigma_1, \sigma_2, s)) \notin Q$
- Divide forgeries in 3 types:
 - Type I: $v^* = v$ and $t^* = t$ (reduce to CR of H)
 - Type II: $v^* = v$ and $t^* \neq t$ (reduce to DLog)
 - Type III: $v^* \neq v$ (reduce to CDH)



PROOF – TYPE I FORGERIES

- sEUF-CMA adversary A outputs $(m^*, (\sigma_1^*, \sigma_2^*, s^*))$ such that $v^* = v$ and $t^* = t$
 - Build adversary B that breaks collision resistance of H
 - Setup: B simply runs setup honestly, and picks H . Output $PK = (g, g^y, g^*, h, u^*, U, H)$ and $SK = (g^*)^y$
 - Signatures: B signs messages honestly
 - Challenge: B receives A's forgery $(m^*, (\sigma_1^*, \sigma_2^*, s^*))$ such that $v^* = v$ corresponding to $(m, (\sigma_1, \sigma_2, s)) \in Q$
 - Analysis: Since $t^* = t$, $t^* = H(M^* || \sigma_2^*)$, $t = H(M || \sigma_2)$, what we want to prove is $M || \sigma_2 \neq M^* || \sigma_2^*$. Say $M = M^*$ and $g^r = \sigma_2 = \sigma_2^*$. We know $v^* = v = H(g^t h^s)$. The fact that $v^* = v$ implies $\sigma_1 = \sigma_1^*$. If $s^* = s$, then A lost. Else, A wins, but produces collision in $H(g^t h^s)$.
- 

PROOF – TYPE II FORGERIES

- sEUF-CMA adversary A outputs $(m^*, (\sigma_1^*, \sigma_2^*, s^*))$ such that $v^* = v$ and $t^* \neq t$
- Build adversary B that breaks Dlog
 - B receives (g, h) from challenger, must find $\log_g h$
 - Setup: inject (g, h) into $PK = (g, g^y, g^*, h, u^*, U, H)$, get $SK = (g^*)^y$ honestly, output PK to A
 - Signature queries: signatures done honestly
 - Forgery: B receives A's forgery $(m^*, (\sigma_1^*, \sigma_2^*, s^*))$ such that $v^* = v$ corresponding to $(m, (\sigma_1, \sigma_2, s)) \in Q$
 - Analysis: As $v^* = v = H(g^t h^s)$, we know $H(g^t h^s) = H(g^{t^*} h^{s^*})$, in which s, s^*, t, t^* are known. Output $a = \frac{t-t^*}{s^*-s}$ as DLog



PROOF – TYPE III FORGERIES

- We will not cover them here.
- Proof is more complicated, and relies on a transformation of EUF-CMA to sEUF-CMA

