Cryptology

Jorge L. Villar

FME, UPC, Fall 2024

Outline

1. Public-Key Encryption
2. PKE Practical Constructions
3. Homomorphic Encryption

Cryptography in a Multiuser Setting

Private Storage: One secret key per user

Private Communication: One shared secret key per channel

In a Private Communication Network
n users $\Rightarrow$ $\binom{n}{2}$ secret keys
Every user stores $n-1$ independent secret keys!

Cryptography in a Multiuser Setting

Private Storage: One secret key per user

Private Communication: One shared secret key per channel

In a Private Communication Network
n users $\Rightarrow$ $\binom{n}{2}$ secret keys
Every user stores $n - 1$ independent secret keys!

Public Key Cryptography: Every user generates a key pair (pk, sk), publishes pk and keeps sk secret

Every user stores only one secret key . . . but public keys must be reliably distributed

The Diffie-Hellman Key Agreement Protocol

The seminal work in public key cryptography!

Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of prime order q.

$$G = \{g^0 = 1, g, g^2, g^3, \ldots, g^{q-1}\}, \qquad g^q = 1$$

E.g. g is a nontrivial solution of $g^q = 1 \pmod{p}$.

p, q are large primes and q divides $p - 1$.

All multiplications and powers are modulo p.

The Diffie-Hellman Key Agreement Protocol

The seminal work in public key cryptography!

Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of prime order q.

$G = \{g^0 = 1, g, g^2, g^3, \ldots, g^{q-1}\}, \qquad g^q = 1$

E.g. g is a nontrivial solution of $g^q = 1 \pmod{p}$.

p, q are large primes and q divides $p - 1$.

All multiplications and powers are modulo p.

Small example (insecure!):
$p = 23$, $q = 11$, $g = 2$ as $2^{11} = 2048 = 23 * 89 + 1$.

The Diffie-Hellman Key Agreement Protocol

Party $P_i(G, q, g)$:
$x_i \leftarrow \{0, \ldots, q-1\}$
publish $y_i = g^{x_i}$
wait for y_j
$k_{ij} = y_j^{x_i}$ // common key for parties P_i and P_j

Common key: $k_{ij} = y_i^{x_j} = y_j^{x_i} = g^{x_i x_j}$.

The Diffie-Hellman Key Agreement Protocol

Party $P_i(G, q, g)$:

$x_i \leftarrow \{0, \ldots, q-1\}$

publish $y_i = g^{x_i}$

wait for y_j

$k_{ij} = y_j^{x_i}$ // common key for parties P_i and P_j

Common key: $k_{ij} = y_i^{x_j} = y_j^{x_i} = g^{x_i x_j}$.

Assumption (Decision Diffie-Hellman (DDH))

(Informal) Given only G, g, y_i and y_j, the common key k_{ij} is indistinguishable from random.

The Diffie-Hellman Key Agreement Protocol

Party $P_i(G, q, g)$:
$x_i \leftarrow \{0, \dots, q-1\}$
publish $y_i = g^{x_i}$
wait for y_j
$k_{ij} = y_j^{x_i}$ // common key for parties P_i and P_j

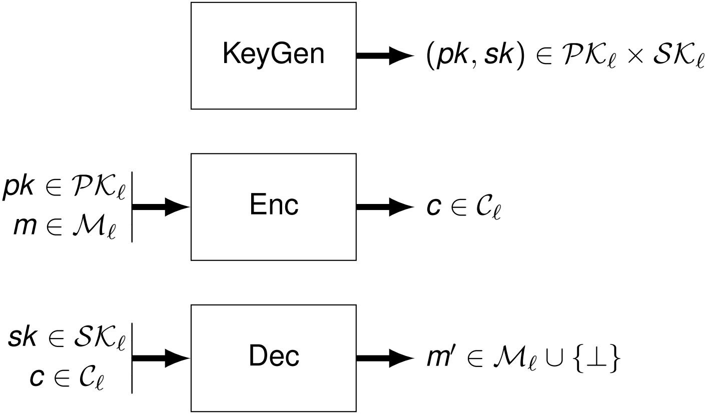
Common key: $k_{ij} = y_i^{x_j} = y_j^{x_i} = g^{x_i x_j}$.

Assumption (Decision Diffie-Hellman (DDH))

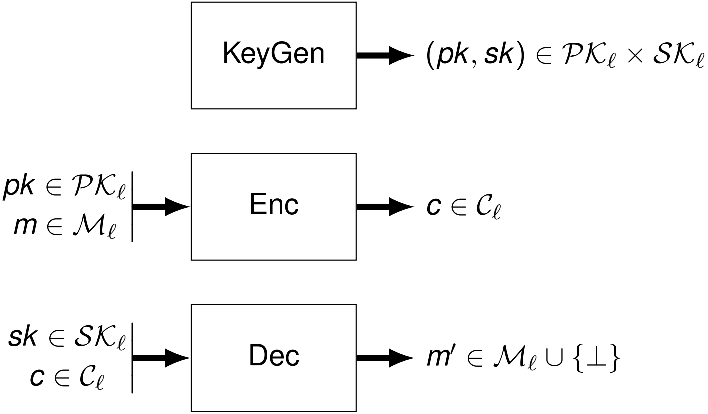
(Informal) Given only G, g, y_i and y_j, the common key k_{ij} is indistinguishable from random.

Combining Diffie-Hellman key agreement protocol with the one-time pad, we can build a **Public-Key Encryption Scheme.**

Public Key Encryption: Syntax



Public Key Encryption: Correctness



$$\forall m \in \mathcal{M}_\ell \ \forall (pk, sk) \leftarrow \mathsf{KeyGen}(\ell), \quad m = \mathsf{Dec}(sk, \mathsf{Enc}(pk, m))$$

The Man-in-the-Middle Attack

A	B
send(B, pk_A);	**receive**(pk_A)
	$c_B \leftarrow \mathrm{Enc}(pk_A, m_B)$;
receive(c_B)	**send**(A, c_B);
$m_B \leftarrow \mathrm{Dec}(sk_A, c_B)$;	

The Man-in-the-Middle Attack

In a multiparty scenario, the adversary C can impersonate a user by replacing the public key

A	C	B
send(B, pk_A);	**intercept**; **send**(B, pk');	**receive**(pk')
receive(c') $m' \leftarrow \mathrm{Dec}(sk_A, c')$;	**intercept**; $m_B \leftarrow \mathrm{Dec}(sk', c_B)$; $c' \leftarrow \mathrm{Enc}(pk_A, m')$; **send**$(A, c')$;	$c_B \leftarrow \mathrm{Enc}(pk', m_B)$; **send**$(A, c_B)$;

The Man-in-the-Middle Attack

Public keys need to be certified

A	C	B
send(B, pk_A);	**intercept**;	
	send(B, pk');	**receive**(pk')
		verify(A, pk');
		$c_B \leftarrow \mathrm{Enc}(pk', m_B)$;
	intercept;	**send**(A, c_B);
	$m_B \leftarrow \mathrm{Dec}(sk', c_B)$;	
	$c' \leftarrow \mathrm{Enc}(pk_A, m')$;	
receive(c')	**send**(A, c');	
$m' \leftarrow \mathrm{Dec}(sk_A, c')$;		

The Man-in-the-Middle Attack

Public keys need to be **certified**

- **Public Key Infrastructure:** Public keys are validated by a trusted entity

The Man-in-the-Middle Attack

Public keys need to be **certified**

- **Public Key Infrastructure:** Public keys are validated by a trusted entity
- **Identity Based Cryptography:** Public keys are just the users' identities, and then they are not replaceable (e.g., they are just the users' names or their email addresses)

The Man-in-the-Middle Attack

Public keys need to be **certified**

- **Public Key Infrastructure:** Public keys are validated by a trusted entity
- **Identity Based Cryptography:** Public keys are just the users' identities, and then they are not replaceable (e.g., they are just the users' names or their email addresses)
  ...but then a Key Generation Center generates and then knows all the secret keys

Outline

ElGamal PKE Scheme

Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of ℓ bits long prime order q. Set $\mathcal{M} = G$ and $\mathcal{C} = G \times G$.

InstGen(ℓ) : // generates some public parameters
  param $\leftarrow (G, q, g)$

KeyGen(param) :
  $x \leftarrow \mathbb{Z}_q$
  $y \leftarrow g^x$
  output (y, x)

Enc(param, y, m) :
  $r \leftarrow \mathbb{Z}_q$ // probabilistic encryption
  output $(g^r, y^r m)$ // Diffie-Hellman Key & One-Time Pad

Dec(param, x, (c_1, c_2)) :
  output $c_2 c_1^{-x}$ // encryption randomness is removed

ElGamal PKE Scheme

Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of ℓ bits long prime order q. Set $\mathcal{M} = G$ and $\mathcal{C} = G \times G$.

InstGen(ℓ) : // generates some public parameters
  param $\leftarrow (G, q, g)$

KeyGen(param) :
  $x \leftarrow \mathbb{Z}_q$
  $y \leftarrow g^x$
  output (y, x)

Enc(param, y, m) :
  $r \leftarrow \mathbb{Z}_q$ // probabilistic encryption
  output $(g^r, y^r m)$ // Diffie-Hellman Key & One-Time Pad

Dec(param, $x, (c_1, c_2)$) :
  output $c_2 c_1^{-x}$ // encryption randomness is removed

Correctness:

$$c_2 c_1^{-x} = y^r m (g^r)^{-x} = g^{rx} m g^{-rx} = m.$$

Hashed ElGamal PKE Scheme

Let $G = \langle g \rangle$ be a cyclic (multiplicative) group of ℓ bits long prime order q. Set $\mathcal{M} = \{0,1\}^\ell$ and $\mathcal{C} = G \times \{0,1\}^\ell$.

InstGen(ℓ) : // generates some public parameters
  param $\leftarrow (G, q, g, H)$ // H is a hash function

KeyGen(param) :
  $x \leftarrow \mathbb{Z}_q$
  $y \leftarrow g^x$
  output (y, x)

Enc(param, y, m) :
  $r \leftarrow \mathbb{Z}_q$ // probabilistic encryption
  output $(g^r, H(y^r) \oplus m)$ // Diffie-Hellman Key & One-Time Pad

Dec(param, x, (c_1, c_2)) :
  output $c_2 \oplus H(c_1^x)$ // encryption randomness is removed

Correctness:

$c_2 \oplus H(c_1^x) = H(y^r) \oplus m \oplus H(g^{rx}) = H(g^{rx}) \oplus m \oplus H(g^{rx}) = m$.

ElGamal Variants and Security

Definition (DLOG Problem)

Given G, q, g and a random element $y \in G$, find x such that $y = g^x$.

If DLOG is easy, ElGamal is not secure (the secret key can be obtained form the public information).

But hardness of DLOG is not enough for security of ElGamal!

ElGamal Variants and Security

Definition (DLOG Problem)

Given G, q, g and a random element $y \in G$, find x such that $y = g^x$.

If DLOG is easy, ElGamal is not secure (the secret key can be obtained form the public information).

But hardness of DLOG is not enough for security of ElGamal!

Known ElGamal implementations:

- Subgroups of $GF(p)^\times$ $|p| \approx 2048$ bits, $|q| \approx 224$ bits
- Subgroups of the group of points of an elliptic curve over $GF(p)$ $|p| \approx 224$ bits, $|q| \approx 224$ bits
- Other less common structures (Jacobian varieties, non-abelian groups, . . .)

ElGamal PKE Scheme (Elliptic Curve Version)

Let $G = \langle B \rangle$ be a cyclic (additive) group of ℓ bits long prime order q. Set $\mathcal{M} = G$ and $\mathcal{C} = G \times G$.

InstGen(ℓ) : // generates some public parameters
  param $\leftarrow (G, q, B)$

KeyGen(param) :
  $x \leftarrow \mathbb{Z}_q$
  $Y \leftarrow xB$
  output (Y, x)

Enc(param, Y, M) :
  $r \leftarrow \mathbb{Z}_q$ // probabilistic encryption
  output $(rB, rY + M)$ // Diffie-Hellman Key & One-Time Pad

Dec(param, $x, (C_1, C_2)$) :
  output $C_2 - xC_1$ // encryption randomness is removed

Correctness:

$$C_2 - xC_1 = rY + M - xrB = rxB + M - xrB = M.$$

RSA PKE Scheme

KeyGen(ℓ) :
  Choose random $\ell/2$ bits long primes p, q
  and $e \geq 3$ coprime with $\phi(n) = (p-1)(q-1)$.
  $n \leftarrow pq$
  Set $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n^\times$.
  $d \leftarrow e^{-1} \mod \operatorname{lcm}(p-1, q-1)$
  output $((n, e), (n, d))$ // factoring n must be hard!

Enc(n, e, m) :
  output $m^e \mod n$

Dec(n, d, c) :
  output $c^d \mod n$

RSA PKE Scheme

KeyGen(ℓ) :
Choose random $\ell/2$ bits long primes p, q
and $e \geq 3$ coprime with $\phi(n) = (p-1)(q-1)$.
$n \leftarrow pq$
Set $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n^\times$.
$d \leftarrow e^{-1} \mod \operatorname{lcm}(p-1, q-1)$
output $((n, e), (n, d))$ // factoring n must be hard!

Enc(n, e, m) :
output $m^e \mod n$

Dec(n, d, c) :
output $c^d \mod n$

Correctness:
$c^d \mod n = (m^e)^d \mod n = m^{ed} \mod n = m$,
because $m^{\phi(n)} \mod n = 1$ (Fermat's little theorem)

RSA Variants and Security

Definition (Integer Factoring Problem)

Given $n = pq$, for p, q primes of the same length, find p or q.

If Factoring is easy, RSA is not secure (the secret key can be obtained form the public information).

Typical size of p and q is 1024 bits.

RSA Variants and Security

Definition (Integer Factoring Problem)

Given $n = pq$, for p, q primes of the same length, find p or q.

If Factoring is easy, RSA is not secure (the secret key can be obtained form the public information).

Typical size of p and q is 1024 bits.

Two main ways to generate the public key:

- The public exponent e is random and coprime with $\phi(n) = (p-1)(q-1)$.
- The public exponent e is fixed and $p-1$ and $q-1$ are coprime with e.
  Typical values $e = 17$ or 65537.

Rabin PKE Scheme

KeyGen(ℓ) :
Choose random $\ell/2$ bits long primes p, q such that $p, q = 3 \mod 4$.
$n \leftarrow pq$
Set $\mathcal{M} = \mathcal{C} = QR_n$. // the set of quadratic residues mod n
output $((n, e), (p, q))$ // factoring n must be hard!

Enc(n, m) :
output $m^2 \mod n$

Dec(p, q, c) :
$m_p = c^{(p+1)/4} \mod p$; $m_q = c^{(q+1)/4} \mod q$
Use Chineese Remainder Theorem to compute m from (m_p, m_q).
output m

Rabin PKE Scheme

KeyGen(ℓ) :
Choose random $\ell/2$ bits long primes p, q such that $p, q = 3 \mod 4$.
$n \leftarrow pq$
Set $\mathcal{M} = \mathcal{C} = QR_n$. // the set of quadratic residues mod n
output $((n, e), (p, q))$ // factoring n must be hard!

Enc(n, m) :
output $m^2 \mod n$

Dec(p, q, c) :
$m_p = c^{(p+1)/4} \mod p$; $m_q = c^{(q+1)/4} \mod q$
Use Chineese Remainder Theorem to compute m from (m_p, m_q).
output m

Correctness:

Since $m \in QR_n$ then $m = x^2 \mod n$

$$m_p = c^{(p+1)/4} = m^{(p+1)/2} = x^{p+1} = x^2 = m \pmod p$$

$$m_q = c^{(q+1)/4} = m^{(q+1)/2} = x^{q+1} = x^2 = m \pmod q$$

Paillier PKE Scheme

KeyGen(ℓ) :
- Choose random $\ell/2$ bits long primes p, q.
- $n \leftarrow pq$
- Set $\mathcal{M} = \mathbb{Z}_n$ and $\mathcal{C} = \mathbb{Z}_{n^2}^\times$.
- $\lambda \leftarrow \text{lcm}(p-1, q-1)$
- **output** $(n, (n, \lambda))$ // factoring n must be hard!

Enc(n, m) :
- $r \leftarrow \mathbb{Z}_n^\times$
- **output** $(1 + mn)r^n \mod n^2$

Dec(n, λ, c) : // RSA with $e = n$ gives another way to decrypt c
- $c' \leftarrow c^\lambda \mod n^2$
- **output** $\frac{c'-1}{n}\lambda^{-1} \mod n$

Paillier PKE Scheme

Correctness:

$c' = c^\lambda \mod n^2 = (1 + mn)^\lambda r^{n\lambda} \mod n^2 = 1 + \lambda mn \mod n^2$.

Then, $\frac{c'-1}{n} = \frac{\lambda mn \mod n^2}{n} = m\lambda \mod n$

and $\frac{c'-1}{n}\lambda^{-1} \mod n = m$.

Paillier PKE Scheme

Correctness:

$c' = c^\lambda \mod n^2 = (1+mn)^\lambda r^{n\lambda} \mod n^2 = 1 + \lambda mn \mod n^2$.
Then, $\frac{c'-1}{n} = \frac{\lambda mn \mod n^2}{n} = m\lambda \mod n$
and $\frac{c'-1}{n}\lambda^{-1} \mod n = m$.

Another way to decrypt: Use RSA with $e = n$

$c^d \mod n = ((1+mn)r^n)^d \mod n = r^{nd} \mod n = r$.

Once r is recovered, then

$\frac{(cr^{-n}-1) \mod n^2}{n} = \frac{((1+mn)-1) \mod n^2}{n} = m$.

Paillier PKE Scheme

Definition (Integer Factoring Problem)

Given $n = pq$, for p, q primes of the same length, find p or q.

If Factoring is easy, Paillier is not secure (the secret key can be obtained form the public information).

Typical size of p and q is 1024 bits.

Paillier PKE Scheme

Definition (Integer Factoring Problem)

Given $n = pq$, for p, q primes of the same length, find p or q.

If Factoring is easy, Paillier is not secure (the secret key can be obtained form the public information).

Typical size of p and q is 1024 bits.

Variant: $1 + nm$ can be replaced by $g^m \mod n^2$, for some special values of g.

Encryption is now $c = g^m r^n \mod n^2$

Decryption uses the fact that $g^\lambda \mod n^2 = 1 + \beta n$ for some β.

Regev PKE

InstGen(ℓ) :
Select a prime q and integers n, k.
Select a Gaussian distribution $N(0, \sigma^2)$.
Set $\mathcal{M} = \{0, 1\}$.
param $\leftarrow (q, n, k, \sigma)$

KeyGen(ℓ) :
$A \leftarrow \mathbb{Z}_q^{k\times n}$; $\mathbf{s} \leftarrow \mathbb{Z}_q^{n\times 1}$; $\mathbf{x} \leftarrow \mathbb{Z}_q^{k\times 1}$;
where $x_i \leftarrow \mathbf{round}(qN(0, \sigma^2)) \mod q$
output$((A, \mathbf{b} = A\mathbf{s} + \mathbf{x} \mod q), \mathbf{s})$

Enc(param, A, $\mathbf{b}$, m) :
$\mathbf{r} \leftarrow \{0, 1\}^{1\times k}$
output $(\mathbf{c}_1, c_2) = (\mathbf{r}A \mod q, \mathbf{r} \cdot \mathbf{b} + m\frac{q-1}{2} \mod q)$

Dec(param, $\mathbf{s}$, $(\mathbf{c}_1, c_2)$) :
$\mu = c_2 - \mathbf{c}_1 \cdot \mathbf{s} \mod q$
Decide m by proximity of μ to $\{0, \frac{q-1}{2}\}$
output m

Regev PKE

Correctness:

$\mu = c_2 - \mathbf{c_1} \cdot \mathbf{s} \mod q = \mathbf{r} \cdot \mathbf{b} + m\frac{q-1}{2} - \mathbf{r}A \cdot \mathbf{s} \mod q =$
$\mathbf{r} \cdot (A\mathbf{s} + \mathbf{x}) + m\frac{q-1}{2} - \mathbf{r}A \cdot \mathbf{s} \mod q = \mathbf{r} \cdot \mathbf{x} + m\frac{q-1}{2} \mod q.$

$\mathbf{r} \cdot \mathbf{x}$ is small (Depends on k and σ) $\Rightarrow$ $\mu \approx m\frac{q-1}{2}$.

There is a (tiny) positive probability of decryption error.

Regev PKE

Correctness:

$\mu = c_2 - \mathbf{c}_1 \cdot \mathbf{s} \mod q = \mathbf{r} \cdot \mathbf{b} + m\frac{q-1}{2} - \mathbf{r}A \cdot \mathbf{s} \mod q =$
$\mathbf{r} \cdot (A\mathbf{s} + \mathbf{x}) + m\frac{q-1}{2} - \mathbf{r}A \cdot \mathbf{s} \mod q = \mathbf{r} \cdot \mathbf{x} + m\frac{q-1}{2} \mod q.$

$\mathbf{r} \cdot \mathbf{x}$ is small (Depends on k and σ) $\Rightarrow$ $\mu \approx m\frac{q-1}{2}$.

There is a (tiny) positive probability of decryption error.

Typical choice for parameters:

- $n^2 < q < 2n^2$, q prime
- $k = (1 + \epsilon)(n + 1) \log q$ for some ϵ
- $\sigma = o(1/\sqrt{n} \log n)$

Security depends on some lattice theory problems (no efficient quantum attack is known!).

Summary of Constructions

- ElGamal is based on DLOG, Regev uses lattices and RSA, Rabin and Paillier are based on Factoring
- ElGamal can be based on elliptic curves for shorter keysizes
- All except Regev are broken with a quantum computer
- RSA and Rabin are deterministic, while the others are probabilistic
- The ratio of ciphertext/plaintext sizes is 1 for RSA and Rabin, 2 for ElGamal and Paillier, and it is huge for Regev

Outline

Homomorphic Encryption

Let $\Pi = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be a PKE scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\otimes$ be a binary operation defined on $\mathcal{M}$.

Definition (Weakly Homomorphic Encryption)

Π is weakly homomorphic with respect to $\otimes$ if there exists a an efficient algorithm $\mathrm{HomEval}(\cdot)$ such that for all properly generated (pk, sk) and all messages $m_1, m_2 \in \mathcal{M}$

$$\mathrm{Dec}(sk, \mathrm{HomEval}(pk, \mathrm{Enc}(pk, m_1), \mathrm{Enc}(pk, m_2))) = m_1 \otimes m_2$$

Homomorphic Encryption

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Let $\otimes$ be a binary operation defined on $\mathcal{M}$.

Definition (Weakly Homomorphic Encryption)

Π is weakly homomorphic with respect to $\otimes$ if there exists a an efficient algorithm $\mathsf{HomEval}(\cdot)$ such that for all properly generated (pk, sk) and all messages $m_1, m_2 \in \mathcal{M}$

$$\mathsf{Dec}(sk, \mathsf{HomEval}(pk, \mathsf{Enc}(pk, m_1), \mathsf{Enc}(pk, m_2))) = m_1 \otimes m_2$$

It means computing on encrypted data without decrypting it!

Examples

Typically, there exist efficient group operations, say $\otimes$ in $\mathcal{M}$ and $\odot$ on $\mathcal{C}$, such that

$$\mathrm{Dec}(sk, c_1 \odot c_2) = \mathrm{Dec}(sk, c_1) \otimes \mathrm{Dec}(sk, c_2)$$

and one can just define $\mathrm{HomEval}(pk, c_1, c_2) = c_1 \odot c_2$ to obtain a weak homomorphic PKE.

Examples

Typically, there exist efficient group operations, say $\otimes$ in $\mathcal{M}$ and $\odot$ on $\mathcal{C}$, such that

$$\mathrm{Dec}(sk, c_1 \odot c_2) = \mathrm{Dec}(sk, c_1) \otimes \mathrm{Dec}(sk, c_2)$$

and one can just define $\mathrm{HomEval}(pk, c_1, c_2) = c_1 \odot c_2$ to obtain a weak homomorphic PKE.

Example: RSA is homomorphic w.r.t. multiplication in $\mathbb{Z}_n^\times$

$\mathrm{Dec}(n, d, c_1 c_2 \mod n) = (c_1 c_2)^d \mod n = c_1^d c_2^d \mod n =$
$= \mathrm{Dec}(n, d, c_1)\mathrm{Dec}(n, d, c_2) \mod n$

or

$\mathrm{Enc}(n, e, m_1)\mathrm{Enc}(n, e, m_2) \mod n = m_1^e m_2^e \mod n =$
$= (m_1 m_2)^e \mod n = \mathrm{Enc}(n, e, m_1 m_2 \mod n)$

Strongly Homomorphic Encryption

Weak homomorphism can leak some information about

- whether a ciphertext c is computed via HomEval or directly with Enc,
- whether certain ciphertext c_1 was used in a call to HomEval.

Strongly Homomorphic Encryption

Weak homomorphism can leak some information about

- whether a ciphertext c is computed via HomEval or directly with Enc,
- whether certain ciphertext c_1 was used in a call to HomEval.

Definition (Strongly Homomorphic Encryption)

Π is strongly homomorphic if in addition, for any valid (pk, sk), m_1 and m_2, the random variables $\mathsf{Enc}(pk, m_1 \otimes m_2)$ and $\mathsf{HomEval}(pk, \mathsf{Enc}(pk, m_1), \mathsf{Enc}(pk, m_2))$ are independent and identically distributed.

Strongly Homomorphic Encryption

Weak homomorphism can leak some information about

- whether a ciphertext c is computed via HomEval or directly with Enc,
- whether certain ciphertext c_1 was used in a call to HomEval.

Definition (Strongly Homomorphic Encryption)

Π is strongly homomorphic if in addition, for any valid (pk, sk), m_1 and m_2, the random variables $\text{Enc}(pk, m_1 \otimes m_2)$ and $\text{HomEval}(pk, \text{Enc}(pk, m_1), \text{Enc}(pk, m_2))$ are independent and identically distributed.

Lemma

Any weakly homomorphic deterministic PKE is also strongly.

Ciphertext Rerandomization

In a strongly homomorphic probabilistic PKE, HomEval(.) needs to remove any correlation between the randomness in $\mathrm{Enc}(pk, m_1)$, $\mathrm{Enc}(pk, m_2)$ and $\mathrm{Enc}(pk, m_1 \otimes m_2)$.

Main idea: Define $\mathrm{ReRand}(pk, \cdot)$, so that it transforms any ciphertext c into another one c' such that

- $\mathrm{Dec}(sk, c) = \mathrm{Dec}(sk, c')$,
- c and c' have independent randomness,
- c' has the same probability distribution as $\mathrm{Enc}(pk, m)$.

Ciphertext Rerandomization

In a strongly homomorphic probabilistic PKE, HomEval(.) needs to remove any correlation between the randomness in $\mathrm{Enc}(pk, m_1)$, $\mathrm{Enc}(pk, m_2)$ and $\mathrm{Enc}(pk, m_1 \otimes m_2)$.

Main idea: Define $\mathrm{ReRand}(pk, \cdot)$, so that it transforms any ciphertext c into another one c' such that

- $\mathrm{Dec}(sk, c) = \mathrm{Dec}(sk, c')$,
- c and c' have independent randomness,
- c' has the same probability distribution as $\mathrm{Enc}(pk, m)$.

E.g., $\mathrm{ReRand}(pk, c) = \mathrm{WeakHomEval}(pk, c, \mathrm{Enc}(pk, m_0))$, where m_0 is the neutral element (0 for addition, 1 for multiplication)

Then, we simply define:

$$\mathrm{HomEval}(pk, c_1, c_2) = \mathrm{ReRand}(pk, \mathrm{WeakHomEval}(pk, c_1, c_2)).$$

Examples

ElGamal is strongly homomorphic in G:

$\mathsf{Enc}(y, m_1) \cdot \mathsf{Enc}(y, m_2) = (g^{r_1}, y^{r_1} m_1) \cdot (g^{r_2}, y^{r_2} m_2) = (g^{r_1+r_2}, y^{r_1+r_2} m_1 m_2)$ is an encryption of $m_1 m_2$.

$\mathsf{HomEval}(y, c_1, c_2) = c_1 \cdot c_2 \cdot (g^{r_3}, y^{r_3})$ for a random $r_3 \in \mathbb{Z}_q$.

It results in $(g^r, y^r m_1 m_2)$ for $r = r_1 + r_2 + r_3$, which is independent of r_1 and r_2.

Examples

Paillier is (additive) strongly homomorphic in $\mathbb{Z}_n$:

$\mathrm{Enc}(n, m_1)\mathrm{Enc}(n, m_2) \mod n^2 = (1 + m_1 n)r_1^n(1 + m_2 n)r_2^n \mod n^2 = (1 + (m_1 + m_2)n)(r_1 r_2)^n \mod n^2$ is an encryption of $m_1 + m_2 \mod n$.

$\mathrm{HomEval}(n, c_1, c_2) = c_1 c_2 r_3^n \mod n^2$ for a random $r_3 \in \mathbb{Z}_n^\times$.

It results in $(1 + (m_1 + m_2)n)r^n \mod n^2$ for $r = r_1 r_2 r_3 \mod n$, which is independent of r_1 and r_2.

Homomorphic Encryption Summary

- RSA is homomorphic w.r.t. multiplication $\mod n$, for $n = pq$.
- ElGamal is homomorphic w.r.t. the operation in the group G.
- Hashed ElGamal has no homomorphic property due to the hash function.
- Paillier is homomorphic w.r.t. addition $\mod n$, for $n = pq$.
- Regev is weakly homomorphic w.r.t. addition $\mod 2$ (XOR operation).

A client application would be interested in common logical or arithmetic operations on bitstrings (none of the above are!).

Applications

If $n = pq$ is large enough, Paillier is quite homomorphic for normal addition of positive integers $\Rightarrow$ Lots of applications!

It can also be done with ElGamal and exponentiated messages $m = g^x$. Then, $g^{x_1} g^{x_2} = g^{x_1+x_2}$, but decryption (i.e. recovering x) can only be done if x is small.

Applications

If $n = pq$ is large enough, Paillier is quite homomorphic for normal addition of positive integers $\Rightarrow$ Lots of applications!

It can also be done with ElGamal and exponentiated messages $m = g^x$. Then, $g^{x_1} g^{x_2} = g^{x_1+x_2}$, but decryption (i.e. recovering x) can only be done if x is small.

Example: (very simplistic) electronic voting

- Encrypted ballot: $c_i = \text{Enc}(pk, ballot)$ where $ballot \in \{0, 1\}$
- Aggregate ballot: $c \leftarrow \text{HomEval}(pk, c_1, \cdots, c_n)$
- Tally computation: $tally = \text{Dec}(sk, c)$, where $tally = \sum_i ballot = \#\{ballot = 1\}$

Applications

If $n = pq$ is large enough, Paillier is quite homomorphic for normal addition of positive integers $\Rightarrow$ Lots of applications!

It can also be done with ElGamal and exponentiated messages $m = g^x$. Then, $g^{x_1} g^{x_2} = g^{x_1+x_2}$, but decryption (i.e. recovering x) can only be done if x is small.

Example: (very simplistic) electronic voting

- Encrypted ballot: $c_i = \mathrm{Enc}(pk, ballot)$ where $ballot \in \{0, 1\}$
- Aggregate ballot: $c \leftarrow \mathrm{HomEval}(pk, c_1, \cdots, c_n)$
- Tally computation: $tally = \mathrm{Dec}(sk, c)$, where $tally = \sum_i ballot = \#\{ballot = 1\}$

Many verifications should be added for a realistic voting scheme (detection of corrupted ballots, multiple voting, corrupted tally. . .) while preserving voter's privacy.

Fully Homomorphic Encryption

A **fully homomorphic** PKE scheme has homomorphic properties for more than one operation (e.g., addition and multiplication).

Fully Homomorphic Encryption

A **fully homomorphic** PKE scheme has homomorphic properties for more than one operation (e.g., addition and multiplication).

No efficient fully homomorphic encryption scheme is known to date. . . but if one exists, then one can securely evaluate polynomial functions (arithmetic circuits) on encrypted data.

Fully Homomorphic Encryption

A **fully homomorphic** PKE scheme has homomorphic properties for more than one operation (e.g., addition and multiplication).

No efficient fully homomorphic encryption scheme is known to date. . . but if one exists, then one can securely evaluate polynomial functions (arithmetic circuits) on encrypted data.

Arithmetic circuits over the ring $\mathbb{Z}_n$ include boolean circuits, because if one encodes logical '0' and '1' as arithmetic '0' and '1', then

- x_1 **AND** $x_2 = x_1 x_2$
- x_1 **OR** $x_2 = x_1 + x_2 - x_1 x_2$
- x_1 **XOR** $x_2 = x_1 + x_2 - 2x_1 x_2$
- **NOT** $x = 1 - x$

Cryptology

Jorge L. Villar

FME, UPC, Fall 2024

END