Public-Key Cryptography
Public-Key Cryptography

Lecture 8
Public-Key Cryptography

Lecture 8

Public-Key Encryption from Trapdoor OWP
Public-Key Cryptography

Lecture 8
Public-Key Encryption from Trapdoor OWP
CCA Security
Abstracting El Gamal

KeyGen: $PK=(G,g,Y), SK=(G,g,y)$

$Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

$Dec_{(G,g,y)}(X,C) = CX^{-y}$
Abstracting El Gamal

KeyGen: PK=(G, g, Y), SK=(G, g, y)

Enc_{(G, g, Y)}(M) = (X=g^x, C=MY^x)

Dec_{(G, g, y)}(X, C) = CX^{-y}
Abstracting El Gamal

Trapdoor PRG:

KeyGen: PK=(G,g,Y), SK=(G,g,y)
Enc_{(G,g,Y)}(M) = (X=g^x, C=MY^x)
Dec_{(G,g,y)}(X,C) = CX^{-y}
Abstracting El Gamal

- **Trapdoor PRG:**
- **KeyGen:** a pair (PK, SK)

KeyGen: (PK, SK)

\[ \text{KeyGen: } PK = (G, g, Y), \ SK = (G, g, y) \]

\[ \text{Enc}_{(G, g, Y)}(M) = (X = g^x, \ C = MY^x) \]

\[ \text{Dec}_{(G, g, y)}(X, C) = CX^{-y} \]
Abstracting El Gamal

- **Trapdoor PRG:**

  - **KeyGen:** a pair (PK, SK)
  - **Three functions:** $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)

- **KeyGen:** PK=$(G, g, Y)$, SK=$(G, g, y)$

- **Enc**$(G, g, Y)(M) = (X=g^x, C=MY^x)$

- **Dec**$(G, g, y)(X, C) = CX^{-y}$

- **KeyGen:** (PK, SK)
Abstracting El Gamal

Trapdoor PRG:

- **KeyGen**: a pair (PK, SK)
- Three functions: $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)

KeyGen: PK=(G,g,Y), SK=(G,g,y)

Enc$(G,g,Y)$(M) = (X=g^x, C=M.Y^x)

Dec$(G,g,y)$(X,C) = C.X^{-y}

KeyGen: (PK,SK)

Enc$_{PK}$(M) = (X=T$_{PK}$(x), C=M.G$_{PK}$(x))
Abstracting El Gamal

- **Trapdoor PRG:**
  - **KeyGen:** a pair \((PK, SK)\)
  - Three functions: \(G_{PK}(.)\) (a PRG) and \(T_{PK}(.)\) (make trapdoor info) and \(R_{SK}(.)\) (opening the trapdoor)

**KeyGen:** \(PK=(G,g,Y), SK=(G,g,y)\)

**Enc**\(_{(G,g,Y)}(M)\) = \((X=g^x, C=MY^x)\)

**Dec**\(_{(G,g,y)}(X,C)\) = \(CX^{-y}\)

**KeyGen:** \((PK,SK)\)

**Enc**\(_{PK}(M)\) = \((X=T_{PK}(x), C=M.G_{PK}(x))\)

**Dec**\(_{SK}(X,C)\) = \(C/R_{SK}(T_{PK}(x))\)
Abstracting El Gamal

**Trapdoor PRG:**

- **KeyGen:** a pair (PK, SK)
- **Three functions:** $G_{PK}(.)$ (a PRG) and $T_{PK}(.)$ (make trapdoor info) and $R_{SK}(.)$ (opening the trapdoor)
- $G_{PK}(x)$ is pseudorandom even given $T_{PK}(x)$ and PK

**KeyGen:** $PK=(G,g,Y), SK=(G,g,y)$

**Enc**$_{(G,g,Y)}(M) = (X=g^x, C=MY^x)$

**Dec**$_{(G,g,y)}(X,C) = CX^{-y}$

**KeyGen:** $(PK,SK)$

**Enc**$_{PK}(M) = (X=T_{PK}(x), C=M.G_{PK}(x))$

**Dec**$_{SK}(X,C) = C/R_{SK}(T_{PK}(x))$
Abstracting El Gamal

- **Trapdoor PRG:**
  - **KeyGen:** a pair (PK,SK)
  - Three functions: \( G_{PK}(.) \) (a PRG) and \( T_{PK}(.) \) (make trapdoor info) and \( R_{SK}(.) \) (opening the trapdoor)
  - \( G_{PK}(x) \) is pseudorandom even given \( T_{PK}(x) \) and PK
  - \( (PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r) \)

- **Encryption and Decryption**
  - **KeyGen:** PK=(G,g,Y), SK=(G,g,y)
  - **Enc\(_{(G,g,Y)}\)(M) = (X=g\(^X\), C=MY\(^X\))
  - **Dec\(_{(G,g,y)}\)(X,C) = CX\(^{-y}\)

- **Encryption**
  - **Enc\(_{PK}\)(M) = (X=T_{PK}(x), C=M.G_{PK}(x))
  - **Dec\(_{SK}\)(X,C) = C/R_{SK}(T_{PK}(x))
Abstracting El Gamal

**Trapdoor PRG:**

- **KeyGen:** a pair (PK, SK)
- **Three functions:** \( G_{PK}(.) \) (a PRG) and \( T_{PK}(.) \) (make trapdoor info) and \( R_{SK}(.) \) (opening the trapdoor)
- \( G_{PK}(x) \) is pseudorandom even given \( T_{PK}(x) \) and PK
- \((PK,T_{PK}(x),G_{PK}(x)) \approx (PK,T_{PK}(x),r)\)
- \( T_{PK}(x) \) hides \( G_{PK}(x) \). SK opens it.

**KeyGen:** PK=(G,g,Y), SK=(G,g,y)
**Enc**(G,g,Y)(M) = \((X=g^x, C=MY^x)\)
**Dec**(G,g,y)(X,C) = \( CX^{-y} \)

**KeyGen:** (PK,SK)
**EncPK(M) = (X=T_{PK}(x), C=M.G_{PK}(x))\)
**DecSK(X,C) = C/R_{SK}(T_{PK}(x))\)
Abstracting El Gamal

- **Trapdoor PRG:**
  - **KeyGen:** a pair \((PK, SK)\)
  - Three functions: \(G_{PK}(.)\) (a PRG) and \(T_{PK}(.)\) (make trapdoor info) and \(R_{SK}(.)\) (opening the trapdoor)
  - \(G_{PK}(x)\) is pseudorandom even given \(T_{PK}(x)\) and \(PK\)
  - \((PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\)
  - \(T_{PK}(x)\) hides \(G_{PK}(x)\). \(SK\) opens it.
  - \(R_{SK}(T_{PK}(x)) = G_{PK}(x)\)

KeyGen: \((PK, SK)\)

\[
\begin{align*}
&\text{Enc}_{PK}(M) = (X=T_{PK}(x), C=M \cdot G_{PK}(x)) \\
&\text{Dec}_{SK}(X, C) = C / R_{SK}(T_{PK}(x))
\end{align*}
\]
Abstracting El Gamal

- **Trapdoor PRG:**
  - **KeyGen:** a pair \((PK, SK)\)
  - Three functions: \(G_{PK}(\cdot)\) (a PRG) and \(T_{PK}(\cdot)\) (make trapdoor info) and \(R_{SK}(\cdot)\) (opening the trapdoor)
    - \(G_{PK}(x)\) is pseudorandom even given \(T_{PK}(x)\) and \(PK\)
    - \((PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\)
    - \(T_{PK}(x)\) hides \(G_{PK}(x)\). \(SK\) opens it.
    - \(R_{SK}(T_{PK}(x)) = G_{PK}(x)\)
  - Enough for an IND-CPA secure PKE scheme

- **KeyGen:** \(PK=(G,g,Y), SK=(G,g,y)\)
  - **Enc\(_{(G,g,Y)}\)(M) = (X=g^x, C=MY^x)\)
  - **Dec\(_{(G,g,Y)}\)(X,C) = CX^{-y}\)

- **Dec\(_{SK}(X,C) = C/R_{SK}(T_{PK}(x))\)

**Diagram:**
- Y \(\leftarrow\) Random y \(Y=g^y\)
- Random x \(\rightarrow\) X
- \(X=g^x\)
- K=\(Y^x\) \(\rightarrow\) C
- C=MK \(\rightarrow\) Random y
- Y=g^y
- KeyGen: \(PK=(G,g,Y), SK=(G,g,y)\)
- **Enc\(_{PK}(M) = (X=T_{PK}(x), C=M.G_{PK}(x))\)
- **Dec\(_{SK}(X,C) = C/R_{SK}(T_{PK}(x))\)
Abstracting El Gamal

- **Trapdoor PRG:**
  - **KeyGen:** a pair \((PK, SK)\)
  - Three functions: \(G_{PK}(\cdot)\) (a PRG) and \(T_{PK}(\cdot)\) (make trapdoor info) and \(R_{SK}(\cdot)\) (opening the trapdoor)
  - \(G_{PK}(x)\) is pseudorandom even given \(T_{PK}(x)\) and \(PK\)
  - \((PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\)
  - \(T_{PK}(x)\) hides \(G_{PK}(x)\). \(SK\) opens it.
  - \(R_{SK}(T_{PK}(x)) = G_{PK}(x)\)

- Enough for an IND-CPA secure PKE scheme (cf. Security of El Gamal)

### KeyGen

- PK = \((G, g, Y)\), SK = \((G, g, y)\)

### Enc and Dec

\[\begin{align*}
\text{Enc}_{(G, g, Y)}(M) &= (X = g^x, C = MY^x) \\
\text{Dec}_{(G, g, y)}(X, C) &= CX^{-y}
\end{align*}\]

### KeyGen

- \((PK, SK)\)

\[\begin{align*}
\text{Enc}_{PK}(M) &= (X = T_{PK}(x), C = M.G_{PK}(x)) \\
\text{Dec}_{SK}(X, C) &= C/R_{SK}(T_{PK}(x))
\end{align*}\]
Trapdoor PRG from Generic Assumption?

KeyGen

\((PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\)
Trapdoor PRG from Generic Assumption?

PRG constructed from OWP (or OWF)

\[(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\]
Trapdoor PRG from Generic Assumption?

- PRG constructed from OWP (or OWF)
- Allows us to instantiate the construction with several candidates

\[(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\]
Trapdoor PRG from Generic Assumption?

- PRG constructed from OWP (or OWF)
- Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?

\[(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\]
Trapdoor PRG from Generic Assumption?

- PRG constructed from OWP (or OWF)
- Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
- Trapdoor property seems fundamentally different: generic OWP may not offer such a property

\[
(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)
\]
Trapdoor PRG from Generic Assumption?

- PRG constructed from OWP (or OWF)
  - Allows us to instantiate the construction with several candidates
- Is there a similar construction for TPRG from OWP?
  - Trapdoor property seems fundamentally different: generic OWP may not offer such a property
- Will start with “Trapdoor OWP”

\[ (PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r) \]
Trapdoor OWP
Trapdoor OWP

$(\text{KeyGen}, f, f')$ (all PPT) is a trapdoor one-way permutation (TOWP) if
(KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if

For all (PK,SK) $\leftarrow$ KeyGen
(KeyGen, f, f') (all PPT) is a trapdoor one-way permutation (TOWP) if

For all (PK, SK) $\leftarrow$ KeyGen

$\mathsf{f}_{PK}$ a permutation
(KeyGen, f, f') (all PPT) is a trapdoor one-way permutation (TOWP) if

- For all $(PK, SK) \leftarrow \text{KeyGen}$
  - $f_{PK}$ a permutation
  - $f'_{SK}$ is the inverse of $f_{PK}$
(KeyGen, f, f') (all PPT) is a trapdoor one-way permutation (TOWP) if

For all (PK, SK) \leftarrow \text{KeyGen}

f_{PK} a permutation

f'_{SK} is the inverse of f_{PK}

For all PPT adversary, probability of success in the TOWP experiment is negligible
(KeyGen, f, f') (all PPT) is a trapdoor one-way permutation (TOWP) if

- For all (PK, SK) ← KeyGen
  - $f_{PK}$ a permutation
  - $f'_{SK}$ is the inverse of $f_{PK}$
- For all PPT adversary, probability of success in the TOWP experiment is negligible
Trapdoor OWP

(KeyGen,f,f') (all PPT) is a trapdoor one-way permutation (TOWP) if

- For all (PK,SK) ←KeyGen
  - $f_{PK}$ a permutation
  - $f'_{SK}$ is the inverse of $f_{PK}$
- For all PPT adversary, probability of success in the TOWP experiment is negligible

Hardcore predicate:

- $B_{PK}$ s.t. $(PK,f_{PK}(x),B_{PK}(x)) \approx (PK,f_{PK}(x),r)$
Trapdoor PRG from Trapdoor OWP

\[(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\]
Same construction as PRG from OWP

\[(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)\]
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG

$(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$
Same construction as PRG from OWP

One bit TPRG

KeyGen same as TOWP’s KeyGen
Same construction as PRG from OWP

One bit TPRG

KeyGen same as TOWP’s KeyGen

\[ G_{PK}(x) := B_{PK}(x). \quad T_{PK}(x) := f_{PK}(x). \quad R_{SK}(y) := G_{PK}(f’_{SK}(y)) \]
Same construction as PRG from OWP

One bit TPRG

KeyGen same as TOWP’s KeyGen

\[ G_{PK}(x) := B_{PK}(x). \quad T_{PK}(x) := f_{PK}(x). \]

\[ R_{SK}(y) := G_{PK}(f'_{SK}(y)) \]

(SK assumed to contain PK)
Trapdoor PRG from Trapdoor OWP

- Same construction as PRG from OWP
- One bit TPRG
  - KeyGen same as TOWP's KeyGen
  - $G_{PK}(x) := B_{PK}(x)$. $T_{PK}(x) := f_{PK}(x)$.
  - $R_{SK}(y) := G_{PK}(f'_{SK}(y))$
  - (SK assumed to contain PK)
- More generally, last permutation output serves as $T_{PK}$

---

Diagram:

- $X$ connected to $T$, $G$, and $R$.
- $T$, $G$, and $R$ connected to $PK$ and $SK$.
- $PK$ and $SK$ connected to each other.
- $(PK, T_{PK}(x), G_{PK}(x)) \approx (PK, T_{PK}(x), r)$.
Same construction as PRG from OWP

One bit TPRG

KeyGen same as TOWP’s KeyGen

\[ G_{PK}(x) := B_{PK}(x). \quad T_{PK}(x) := f_{PK}(x). \]

\[ R_{SK}(y) := G_{PK}(f'_{SK}(y)) \]

(SK assumed to contain PK)

More generally, last permutation output serves as \( T_{PK} \)
Same construction as PRG from OWP
One bit TPRG

KeyGen same as TOWP’s KeyGen

\( G_{\text{PK}}(x) := B_{\text{PK}}(x) \). \( T_{\text{PK}}(x) := f_{\text{PK}}(x) \).
\( R_{\text{SK}}(y) := G_{\text{PK}}(f'_{\text{SK}}(y)) \)

(SK assumed to contain PK)

More generally, last permutation output serves as \( T_{\text{PK}} \)
Candidate TOWPs
Candidate TOWPs

From some (candidate) OWP collections, with index as public-key
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - Rabin OWF: $f_{\text{Rabin}}(x; N) = x^2 \mod N$, where $N = PQ$, and $P$, $Q$ are $k$-bit primes (and $x$ uniform from $\{0...N\}$)
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections

  **Rabin OWF:** \( f_{\text{Rabin}}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \{0...N\})

  **Fact:** \( f_{\text{Rabin}}(.; N) \) is a permutation among quadratic residues, when \( P, Q \) are \( \equiv 3 \) (mod 4)
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections

**Rabin OWF**: \( f_{Rabin}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \( \{0...N\} \))

- **Fact**: \( f_{Rabin}(.; N) \) is a permutation among quadratic residues, when \( P, Q \equiv 3 \pmod{4} \)
- **Fact**: Can invert \( f_{Rabin}(.; N) \) given factorization of \( N \)
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - **Rabin OWF**: \( f_{\text{Rabin}}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \{0...N\})
    - **Fact**: \( f_{\text{Rabin}}(.; N) \) is a permutation among quadratic residues, when \( P, Q \equiv 3 \pmod{4} \)
    - **Fact**: Can invert \( f_{\text{Rabin}}(.; N) \) given factorization of \( N \)
  - **RSA function**: \( f_{\text{RSA}}(x; N,e) = x^e \mod N \) where \( N=PQ \), \( P,Q \) \( k \)-bit primes, \( e \) s.t. \( \gcd(e,\varphi(N)) = 1 \) (and \( x \) uniform from \{0...N\})
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWF collections
  - **Rabin OWF**: \( f_{\text{Rabin}}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \( \{0...N\} \))
    - **Fact**: \( f_{\text{Rabin}}(.; N) \) is a permutation among quadratic residues, when \( P, Q \equiv 3 \pmod{4} \)
    - **Fact**: Can invert \( f_{\text{Rabin}}(.; N) \) given factorization of \( N \)
  - **RSA function**: \( f_{\text{RSA}}(x; N,e) = x^e \mod N \) where \( N= PQ \), \( P,Q \) \( k \)-bit primes, \( e \) s.t. \( \gcd(e,\phi(N)) = 1 \) (and \( x \) uniform from \( \{0...N\} \))
    - **Fact**: \( f_{\text{RSA}}(.; N,e) \) is a permutation
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key
- Recall candidate OWP collections

**Rabin OWF:** \( f_{\text{Rabin}}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \( \{0,...,N\} \))

- **Fact:** \( f_{\text{Rabin}}(\cdot; N) \) is a permutation among quadratic residues, when \( P, Q \equiv 3 \pmod{4} \)
- **Fact:** Can invert \( f_{\text{Rabin}}(\cdot; N) \) given factorization of \( N \)

**RSA function:** \( f_{\text{RSA}}(x; N,e) = x^e \mod N \) where \( N= PQ \), \( P,Q \) \( k \)-bit primes, \( e \) s.t. \( \gcd(e,\varphi(N)) = 1 \) (and \( x \) uniform from \( \{0,...,N\} \))

- **Fact:** \( f_{\text{RSA}}(\cdot; N,e) \) is a permutation
- **Fact:** While picking \( (N,e) \), can also pick \( d \) s.t. \( x^{ed} = x \)
Candidate TOWPs

- From some (candidate) OWP collections, with index as public-key

Recall candidate OWF collections

- **Rabin OWF**: \( f_{\text{Rabin}}(x; N) = x^2 \mod N \), where \( N = PQ \), and \( P, Q \) are \( k \)-bit primes (and \( x \) uniform from \( \{0...N\} \))
  - **Fact**: \( f_{\text{Rabin}}(.; N) \) is a permutation among quadratic residues, when \( P, Q \equiv 3 \pmod{4} \)
  - **Fact**: Can invert \( f_{\text{Rabin}}(.; N) \) given factorization of \( N \)

- **RSA function**: \( f_{\text{RSA}}(x; N,e) = x^e \mod N \) where \( N=PQ \), \( P,Q \) \( k \)-bit primes, \( e \) s.t. \( \gcd(e,\phi(N)) = 1 \) (and \( x \) uniform from \( \{0...N\} \))
  - **Fact**: \( f_{\text{RSA}}(.; N,e) \) is a permutation
  - **Fact**: While picking \( (N,e) \), can also pick \( d \) s.t. \( x^{ed} = x \)
Recap
Recap

- CPA-secure PKE
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
  - With a secret-key, trapdoor information can also yield the pseudorandom string
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
  - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
    - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
  - With a secret-key, invert the OWP
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- **Trapdoor PRG**
  - Abstracts what DDH gives for El Gamal
  - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- **Trapdoor OWP**
  - With a secret-key, invert the OWP
  - Can be used to construct Trapdoor PRG
Recap

- CPA-secure PKE
- DH Key-exchange, El Gamal and DDH assumption
- Trapdoor PRG
  - Abstracts what DDH gives for El Gamal
  - With a secret-key, trapdoor information can also yield the pseudorandom string
  - Can be used to get IND-CPA secure PKE scheme
- Trapdoor OWP
  - With a secret-key, invert the OWP
  - Can be used to construct Trapdoor PRG
- Next: CCA secure PKE
CCA Secure PKE
CCA Secure PKE

In SKE, to get CCA security, we used a MAC
CCA Secure PKE

- In SKE, to get CCA security, we used a MAC
- Bob would accept only messages from Alice
CCA Secure PKE

In SKE, to get CCA security, we used a MAC

Bob would accept only messages from Alice

But in PKE, Bob wants to receive messages from Eve as well
CCA Secure PKE

In SKE, to get CCA security, we used a MAC

Bob would accept only messages from Alice

But in PKE, Bob wants to receive messages from Eve as well

Only if it is indeed Eve's own message: she should know her own message!
Chosen Ciphertext Attack
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

A subtle e-mail attack
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

A subtle e-mail attack
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

A subtle e-mail attack
Suppose Enc SIM-CPA secure

Alice → Bob: Enc(m)

A subtle e-mail attack

I look around
for your eyes shining
I seek you
in everything...
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

Alice $\rightarrow$ Bob: Enc(m)

A subtle e-mail attack

I look around for your eyes shining
I seek you in everything...
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

Alice → Bob: Enc(m)

I look around for your eyes shining
I seek you in everything...

A subtle e-mail attack
Chosen Ciphertext Attack

Suppose $\text{Enc}$ SIM-CPA secure

Alice $\rightarrow$ Bob: $\text{Enc}(m)$
Eve: $\text{Hack}(\text{Enc}(m)) = \text{Enc}(m^*)$
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

Alice → Bob: Enc(m)
Eve: Hack(Enc(m)) = Enc(m*)
(where m* = Reverse of m)
Chosen Ciphertext Attack

- Suppose $\text{Enc}$ SIM-CPA secure
- Suppose encrypts a character at a time (still secure)

Alice $\rightarrow$ Bob: $\text{Enc}(m)$

Eve: $\text{Hack}(\text{Enc}(m)) = \text{Enc}(m^*)$
(where $m^* = \text{Reverse of } m$)

I look around for your eyes shining
I seek you in everything...
Chosen Ciphertext Attack

Suppose $\text{Enc}$ SIM-CPA secure

Suppose encrypts a character at a time (still secure)

Alice $\rightarrow$ Bob: $\text{Enc}(m)$
Eve: $\text{Hack}(\text{Enc}(m)) = \text{Enc}(m^*)$
(where $m^*$ = Reverse of $m$)
Eve $\rightarrow$ Bob: $\text{Enc}(m^*)$
Chosen Ciphertext Attack

Suppose $Enc$ SIM-CPA secure

Suppose encrypts a character at a time (still secure)

Alice $\rightarrow$ Bob: $Enc(m)$
Eve: $\forall Enc(m) = Enc(m^*)$
(\textit{where }$m^*$\textit{ = Reverse of }$m$)
Eve $\rightarrow$ Bob: $Enc(m^*)$
Chosen Ciphertext Attack

- Suppose Enc SIM-CPA secure
- Suppose encrypts a character at a time (still secure)

Alice → Bob: Enc(m)
Eve: \( \text{Hack(Enc(m))} = \text{Enc(m^*)} \)
(\( m^* = \text{Reverse of m} \))
Eve → Bob: Enc(m^*)
Bob → Eve: “what’s this: m*?”

Hey Eve,
What’s this that you sent me?

> ...gnihtyreve ni
> uoy kees l
> gnnihls seye ruoy rof
> dnuora kool l

I look around for your eyes shining
I seek you in everything...
Chosen Ciphertext Attack

Suppose Enc SIM-CPA secure

Suppose encrypts a character at a time (still secure)

Alice → Bob: Enc(m)
Eve:  Hack(Enc(m)) = Enc(m*)
    (where m* = Reverse of m)
Eve → Bob: Enc(m*)
Bob → Eve: “what’s this: m*?”
Eve: Reverse m* to find m!

Hey Eve,
What’s this that you sent me?

> ...gnihtyreve ni
> uoy kees I
> gninihs seye ruoy rof
> dnuora kool I
Malleability
Malleability

- Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message

E.g.: Malleability of El Gamal
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message

E.g.: Malleability of El Gamal

Recall: \( \text{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x) \)
Malleability

Malleability: Eve can "malleate" a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a "related" message

E.g.: Malleability of El Gamal

Recall: $\text{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$

Given $(X,C)$ change it to $(X,TC)$: will decrypt to TM
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message.

E.g.: Malleability of El Gamal

Recall: $\text{Enc}_{(G,g,Y)}(m) = (g^x, M \cdot Y^x)$

Given $(X,C)$ change it to $(X, TC)$: will decrypt to $TM$

Or change $(X,C)$ to $(X^a, C^a)$: will decrypt to $M^a$
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message

E.g.: Malleability of El Gamal

Recall: $\text{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$

Given $(X,C)$ change it to $(X,TC)$: will decrypt to $TM$

Or change $(X,C)$ to $(X^a, C^a)$: will decrypt to $M^a$

If chosen-ciphertext attack possible
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message

E.g.: Malleability of El Gamal

Recall: $\text{Enc}_{(G,g,Y)}(m) = (g^x, M.Y^x)$

Given $(X,C)$ change it to $(X,TC)$: will decrypt to $TM$

Or change $(X,C)$ to $(X^a, C^a)$: will decrypt to $M^a$

If chosen-ciphertext attack possible

i.e., Eve can get a ciphertext of her choice decrypted
Malleability

Malleability: Eve can “malleate” a ciphertext (without having to decrypt it) to produce a new ciphertext that would decrypt to a “related” message

E.g.: Malleability of El Gamal

Recall: \( \text{Enc}_{(G,g,Y)}(m) = (g^x,M.Y^x) \)

Given \((X,C)\) change it to \((X,TC)\): will decrypt to TM

Or change \((X,C)\) to \((X^a,C^a)\): will decrypt to \(M^a\)

If chosen-ciphertext attack possible

i.e., Eve can get a ciphertext of her choice decrypted

Then Eve can exploit malleability to learn something “related to” Alice’s messages
Hey Eve,
What's this that you sent me?

Hello Eve,

I look around for your eyes shining
I seek you in everything...

I look around for your eyes shining
I seek you in everything...

Hey Eve,
What's this that you sent me?

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I
Chosen Ciphertext Attack

SIM-CCA: does capture this attack

Hey Eve,
What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

...gnihtyreve ni uoy kees I
gninhs seye ruoy rof
dnuora kool I

Hey Eve,
What's this that you sent me?

...gnihtyreve ni uoy kees I
gninhs seye ruoy rof
dnuora kool I
Hey Eve,

What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

...gnihtyreve ni uoy kees I gniihns seye ruoy rof dnuora kool I

Hey Eve,

What's this that you sent me?
Hey Eve,

What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

Hey Eve,

What's this that you sent me?

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I
Hey Eve,
What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

Hey Eve,
What's this that you sent me?

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I
Hey Eve,

What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

Chosen Ciphertext Attack

SIM-CCA: does capture this attack
Hey Eve,
What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

Chosen Ciphertext Attack

SIM-CCA: does capture this attack

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I

Hey Eve,
What's this that you sent me?
Chosen Ciphertext Attack

SIM-CCA: does capture this attack

Hey Eve,

What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

I look around for your eyes shining
I seek you in everything...

Hey Eve,

What's this that you sent me?

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I
Hey Eve,
What's this that you sent me?

I look around for your eyes shining
I seek you in everything...

Hey Eve,
What's this that you sent me?

...gnihtyreve ni uoy kees I gninihs seye ruoy rof dnuora kool I
SIM-CCA Security (PKE)

Secure (and correct) if:
\[ \forall \exists \text{s.t.} \forall \text{output of is distributed identically in REAL and IDEAL} \]
SIM-CCA Security and Malleability
SIM-CCA Security and Malleability

If can cause Bob to output a message

IDEAL

REAL

Replay Filter
SIM-CCA Security and Malleability

If an adversary can cause Bob to output a message, then it can send such a message to Bob by itself.
SIM-CCA Security and Malleability

If $\mathcal{A}$ can cause Bob to output a message, then $\mathcal{A}$ can send such a message to Bob by itself.

Hence message not a result of malleating.
Constructing CCA Secure PKEs
Constructing CCA Secure PKEs

- Possible from \textit{generic assumptions}
Constructing CCA Secure PKEs

Possible from generic assumptions

e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...
Constructing CCA Secure PKEs

Possible from *generic assumptions*

- e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ... 

- e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency
Constructing CCA Secure PKEs

Possible from generic assumptions

- e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...

- e.g. Using a CPA secure PKE to create two ciphertexts and a “Non-Interactive Zero Knowledge proof” of consistency

- e.g. Include a “NIZK proof of knowledge” of the plaintext
Constructing CCA Secure PKEs

Possible from generic assumptions

- e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...

- e.g. Using a CPA secure PKE to create two ciphertexts and a “Non-Interactive Zero Knowledge proof” of consistency

- e.g. Include a “NIZK proof of knowledge” of the plaintext

Much more efficient from specific number theoretic/algebraic assumptions
Constructing CCA Secure PKEs

Possible from generic assumptions

- e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...

- e.g. Using a CPA secure PKE to create two ciphertexts and a "Non-Interactive Zero Knowledge proof" of consistency

- e.g. Include a "NIZK proof of knowledge" of the plaintext

Much more efficient from specific number theoretic/algebraic assumptions

Even more efficient in the "Random Oracle Model"
Constructing CCA Secure PKEs

Possible from generic assumptions

- e.g. Enhanced T-OWP, Lossy T-OWF, Correlation-secure T-OWF, Adaptive T-OWF/relation, ...

- e.g. Using a CPA secure PKE to create two ciphertexts and a “Non-Interactive Zero Knowledge proof” of consistency

- e.g. Include a “NIZK proof of knowledge” of the plaintext

Much more efficient from specific number theoretic/algebraic assumptions

- Even more efficient in the “Random Oracle Model”

- Significant efficiency gain using “Hybrid Encryption”
CCA Secure PKE: Cramer-Shoup
CCA Secure PKE:
Cramer-Shoup

El Gamal-like: Based on DDH assumption
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\text{QR}_p^*$ for safe prime $p$)
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”
- $\text{Enc}(M) = (C, S)$
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”
- $\text{Enc}(M) = (C, S)$
  - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”
  - $\text{Enc}(M) = (C, S)$
  - $C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$
- $g_1$, $g_2$, $Y$, $W$, $Z$ are part of PK
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an "integrity tag"

$\text{Enc}(M) = (C, S)$

$C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$

$g_1, g_2, Y, W, Z$ are part of PK

$Y = g_1^{y_1} g_2^{y_2}, W = g_1^{w_1} g_2^{w_2}, Z = g_1^{z_1} g_2^{z_2}$. SK contains $(y_1, y_2, w_1, w_2, z_1, z_2)$
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”

Encryption:

$\text{Enc}(M) = (C, S)$

$C = (g_1^x, g_2^x, MY^x)$ and $S = (WZ^{H(C)})^x$

$g_1, g_2, Y, W, Z$ are part of PK

- $Y = g_1^{y_1} g_2^{y_2}$, $W = g_1^{w_1} g_2^{w_2}$, $Z = g_1^{z_1} g_2^{z_2}$

SK contains $(y_1, y_2, w_1, w_2, z_1, z_2)$

Multiple SKs can explain the same PK (unlike El Gamal)
CCA Secure PKE: Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)
- Uses a collision-resistant hash function inside an “integrity tag”

\[ \text{Enc}(M) = (C, S) \]

\[ C = (g_1^x, g_2^x, MY^x) \quad \text{and} \quad S = (WZ^{H(C)})^x \]

- $g_1$, $g_2$, $Y$, $W$, $Z$ are part of PK

- $Y = g_1^{y_1} g_2^{y_2}$, $W = g_1^{w_1} g_2^{w_2}$, $Z = g_1^{z_1} g_2^{z_2}$.

- SK contains $(y_1, y_2, w_1, w_2, z_1, z_2)$

- Trapdoor: Using SK, and $(g_1^x, g_2^x)$ can find $Y^x$, $W^x$, $Z^x$

Multiple SKs can explain the same PK (unlike El Gamal)
CCA Secure PKE:

Cramer-Shoup

- El Gamal-like: Based on DDH assumption
- Uses a prime-order group (e.g., QR\(_p^*\) for safe prime p)
- Uses a collision-resistant hash function inside an “integrity tag”

**Enc(M) = (C, S)**

- \(C = (g_1^x, g_2^x, MY^x)\) and \(S = (WZ^{H(C)})^x\)
- \(g_1, g_2, Y, W, Z\) are part of PK
  - \(Y = g_1^{y_1} g_2^{y_2}, W = g_1^{w_1} g_2^{w_2}, Z = g_1^{z_1} g_2^{z_2}\)
  - SK contains \((y_1, y_2, w_1, w_2, z_1, z_2)\)

- Trapdoor: Using SK, and \((g_1^x, g_2^x)\) can find \(Y^x, W^x, Z^x\)
- If \((g_1^{x_1}, g_2^{x_2}), x_1 \neq x_2\), then “\(Y^x, W^x, Z^x\)” vary with different SKs

Multiple SKs can explain the same PK (unlike El Gamal)
CCA Secure PKE: Cramer-Shoup

El Gamal-like: Based on DDH assumption

Uses a prime-order group (e.g., $\mathbb{QR}_p^*$ for safe prime $p$)

Uses a collision-resistant hash function inside an “integrity tag”

$\text{Enc}(M) = (C, S)$

\[ C = (g_1^x, g_2^x, MY^x) \quad \text{and} \quad S = (WZ^{H(C)})^x \]

$g_1, g_2, Y, W, Z$ are part of PK

\[ Y = g_1^{y_1} g_2^{y_2}, \quad W = g_1^{w_1} g_2^{w_2}, \quad Z = g_1^{z_1} g_2^{z_2}. \]

SK contains $(y_1, y_2, w_1, w_2, z_1, z_2)$

Trapdoor: Using SK, and $(g_1^x, g_2^x)$ can find $Y^x, W^x, Z^x$

If $(g_1^{x_1}, g_2^{x_2}), x_1 \neq x_2$, then “$Y^x, W^x, Z^x$” vary with different SKs

Decryption: Check $S$ (assuming $x_1 = x_2$) and extract $M$
Security of CS Scheme: Proof Sketch
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that
- It contains no information about the message (given just PK)
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that:
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption.
Security of CS Scheme: Proof Sketch

An “invalid encryption” can be used for challenge such that
- It contains no information about the message (given just PK)
- Is indistinguishable from valid encryption, under DDH assumption

But adversary could get information about the specific SK from decryption queries?

\[(g_1, g_1^{x_1}, g_2, g_2^{x_2})\text{ is of the form } (g, g^x, g^y, g^{xy}) \text{ iff } x_1 = x_2\]
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that:
  - It contains no information about the message (given just PK).
  - Is indistinguishable from valid encryption, under DDH assumption.
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK).
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that:
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)
  - Adversary can’t create new “invalid ciphertexts” that get past the integrity check (except with negligible probability)
Security of CS Scheme: Proof Sketch

- An “invalid encryption” can be used for challenge such that
  - It contains no information about the message (given just PK)
  - Is indistinguishable from valid encryption, under DDH assumption
- But adversary could get information about the specific SK from decryption queries?
  - By querying decryption with only valid ciphertexts, adversary gets no information about SK (beyond given by PK)
  - Adversary can’t create new “invalid ciphertexts” that get past the integrity check (except with negligible probability)
  - Any new invalid ciphertext can fool at most a negligible fraction of the possible SKs: so the probability of adversary fooling the specific one used is negligible
Security of CS Scheme: Proof Sketch

An “invalid encryption” can be used for challenge such that
- It contains **no information** about the message (**given just PK**)
- Is **indistinguishable** from valid encryption, **under DDH assumption**

But adversary could get information about the specific SK from decryption queries?
- By querying decryption with only valid ciphertexts, adversary gets **no information** about SK (beyond given by PK)
- **Adversary can’t create new “invalid ciphertexts”** that get past the integrity check (except with negligible probability)
- Any new invalid ciphertext can fool at most a negligible fraction of the possible SKs: so the probability of adversary fooling the specific one used is negligible

Formally using “hybrid argument” (0 advantage in last hybrid)