MIT 6.875

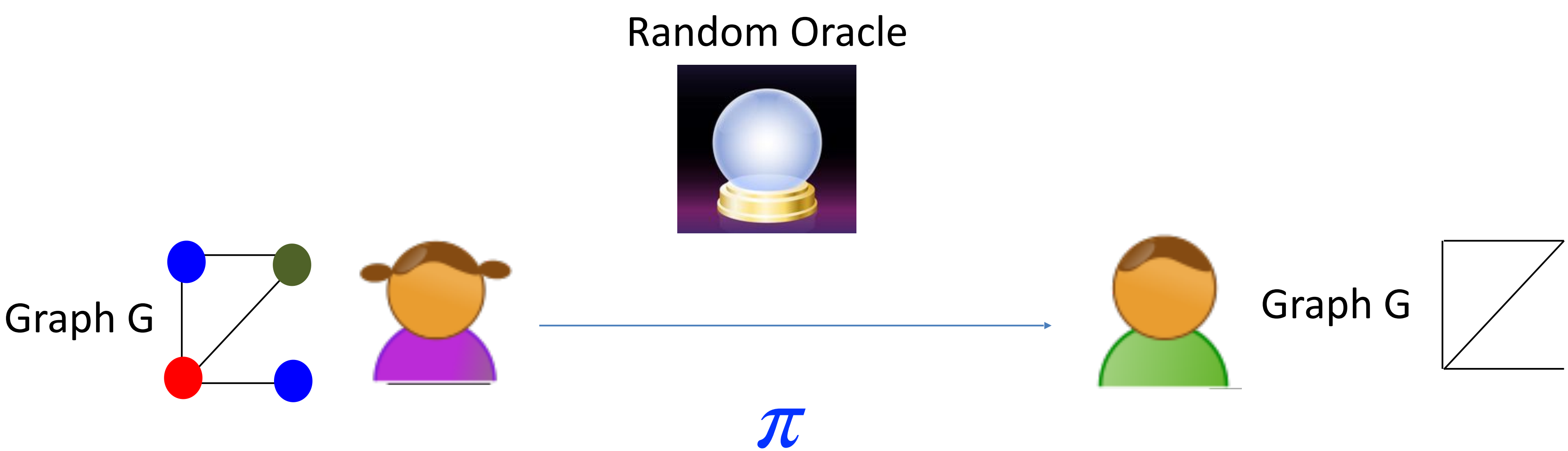
Foundations of Cryptography
Lecture 16

Interaction is Necessary for ZK

Theorem: If a language L has a non-interactive (one-message) ZK proof system, then L can be decided in probabilistic polynomial time.

Two Roads to Non-Interactive ZK (NIZK)

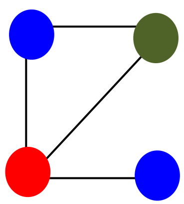
1. Random Oracle Model & Fiat-Shamir Transform.



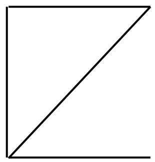
2. Common Random String Model.

The Common Random String Model

Graph G

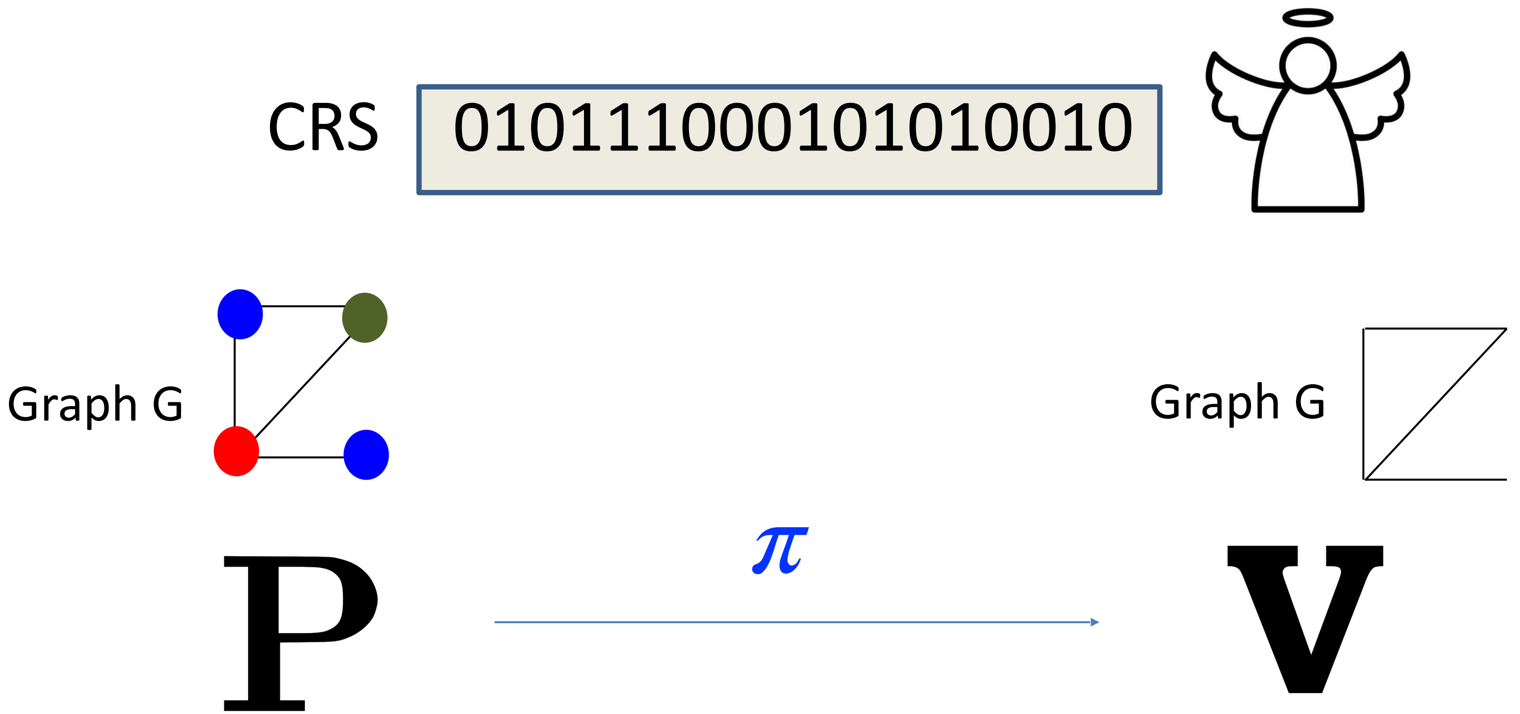


Graph G



P $\xrightarrow{\pi}$ **V**

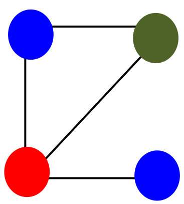
The Common Random String Model



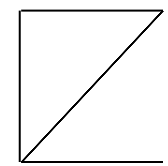
NIZK in the CRS Model

CRS 01011100010101010010

Graph G



Graph G



π

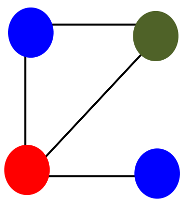
P → **V**

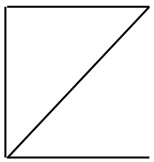
1. **Completeness:** For every $G \in$ 3COL, V accepts P's proof.

2. **Soundness:** For every $G \notin$ 3COL and any "proof" π^*, $V(CRS, \pi^*)$ accepts with probability $\leq \mathrm{neg}(n)$

NIZK in the CRS Model

CRS 010111000101010010

Graph G 

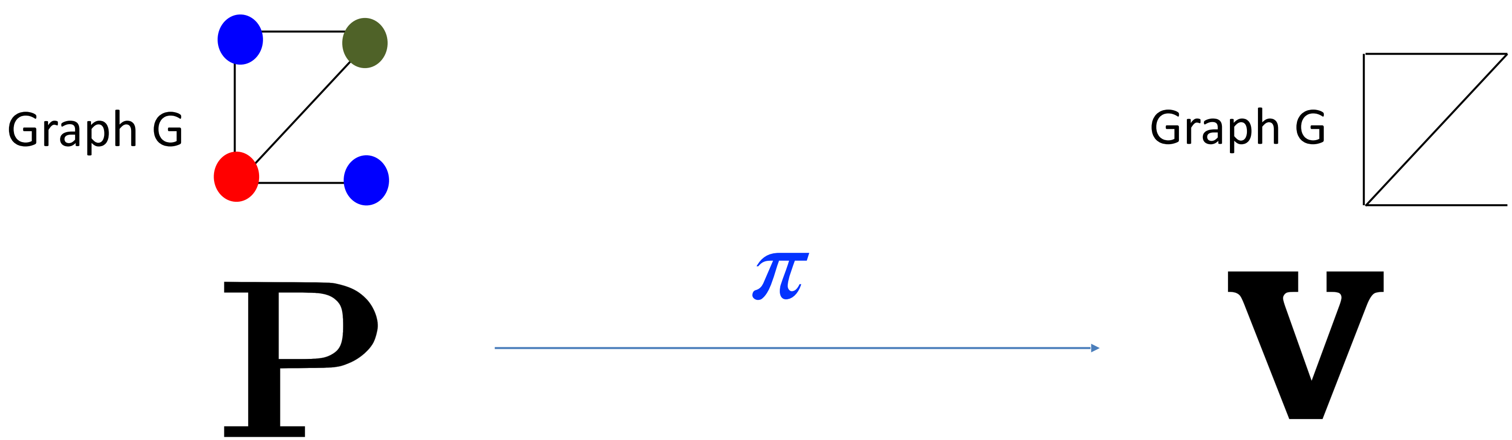
Graph G 

π

P $\longrightarrow$ **V**

NIZK in the CRS Model

CRS 010111000101010010

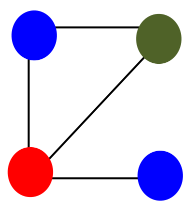


3. Zero Knowledge: There is a PPT simulator S such that for every $G \in$ 3COL, S ***simulates the view*** of the verifier V.

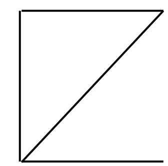
NIZK in the CRS Model

CRS 0101110001010100010

Graph G



Graph G



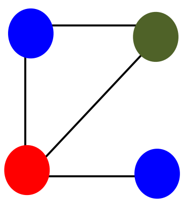
P $\xrightarrow{\pi}$ **V**

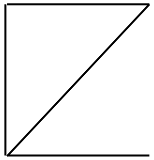
3. Zero Knowledge: There is a PPT simulator S such that for every $G \in$ 3COL, S ***simulates the view*** of the verifier V.

$$S(G) \approx (CRS \leftarrow D,\ \pi \leftarrow P(G, colors))$$

NIZK in the CRS Model

CRS 01011100010101010

Graph G 

Graph G 

P $\xrightarrow{\pi}$ **V**

3. Zero Knowledge: There is a PPT simulator S such that for every $x \in \mathrm{L}$ and witness w, S ***simulates the view*** of the verifier V.

$$S(x) \approx (CRS \leftarrow D,\ \pi \leftarrow P(x, w))$$

HOW TO CONSTRUCT NIZK IN THE CRS MODEL

1. Blum-Feldman-Micali'88 *(quadratic residuosity)*
2. Feige-Lapidot-Shamir'90 *(factoring)*
3. Groth-Ostrovsky-Sahai'06 *(bilinear maps)*
4. Canetti-Chen-Holmgren-Lombardi-Rothblum^2-Wichs'19 and Peikert-Shiehian'19 *(learning with errors)*

HOW TO CONSTRUCT NIZK IN THE CRS MODEL

1. **Blum-Feldman-Micali'88** ***(quadratic residuosity)***

2. Feige-Lapidot-Shamir'90 *(factoring)*

3. Groth-Ostrovsky-Sahai'06 *(bilinear maps)*

4. Canetti-Chen-Holmgren-Lombardi-Rothblum^2-Wichs'19 and Peikert-Shiehian'19 *(learning with errors)*

HOW TO CONSTRUCT NIZK IN THE CRS MODEL

Step 1. **Review** our number theory hammers
& polish them.

Step 2. **Construct** NIZK for a special NP language, namely quadratic *non*-residuosity.

Step 3. **Bootstrap** to NIZK for 3SAT, an NP-complete language.

HOW TO CONSTRUCT NIZK IN THE CRS MODEL

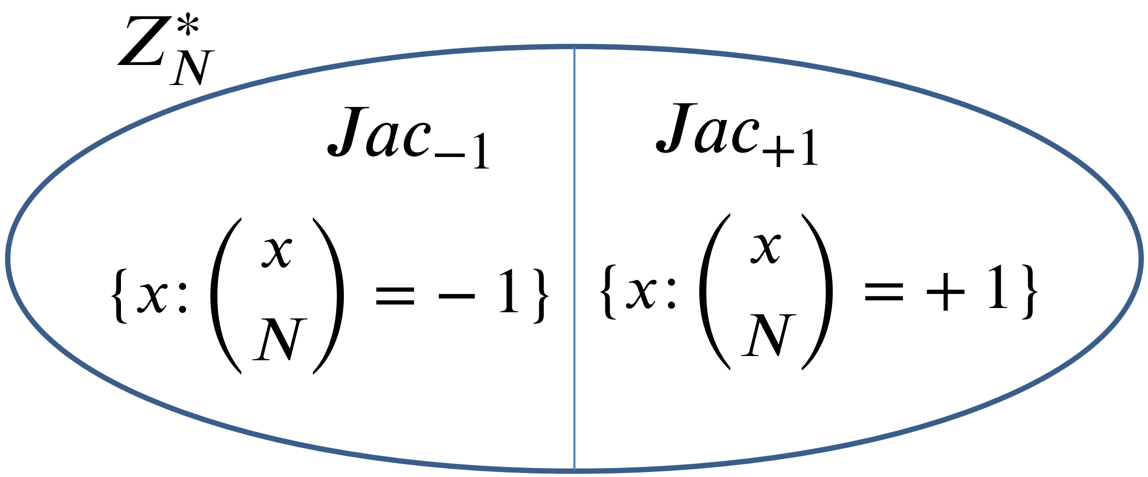
Step 1. **Review** our number theory hammers & polish them.

Step 2. **Construct** NIZK for a special NP language, namely quadratic *non*-residuosity.

Step 3. **Bootstrap** to NIZK for 3SAT, an NP-complete language.

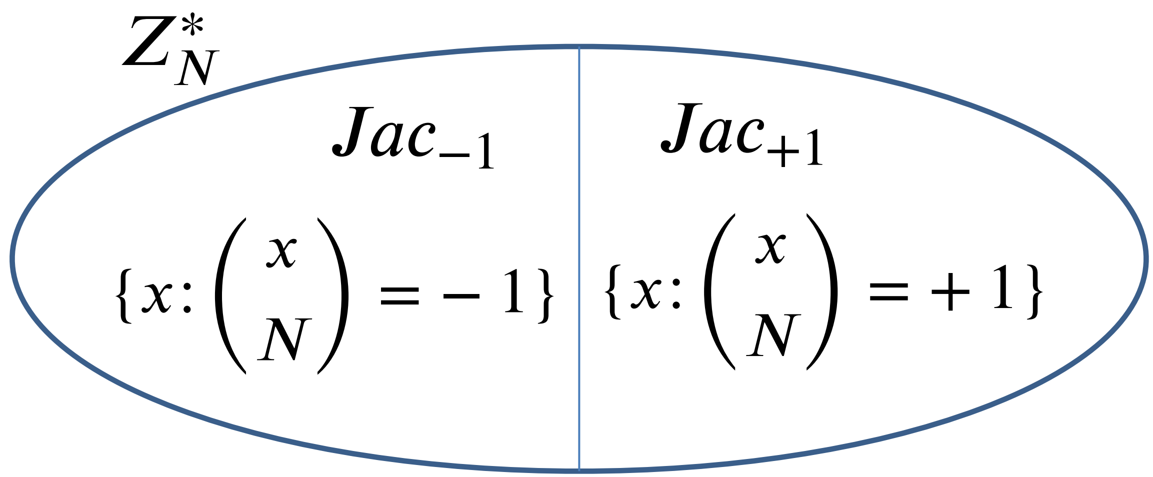
Quadratic Residuosity

Let $N = pq$ be a product of two large primes.



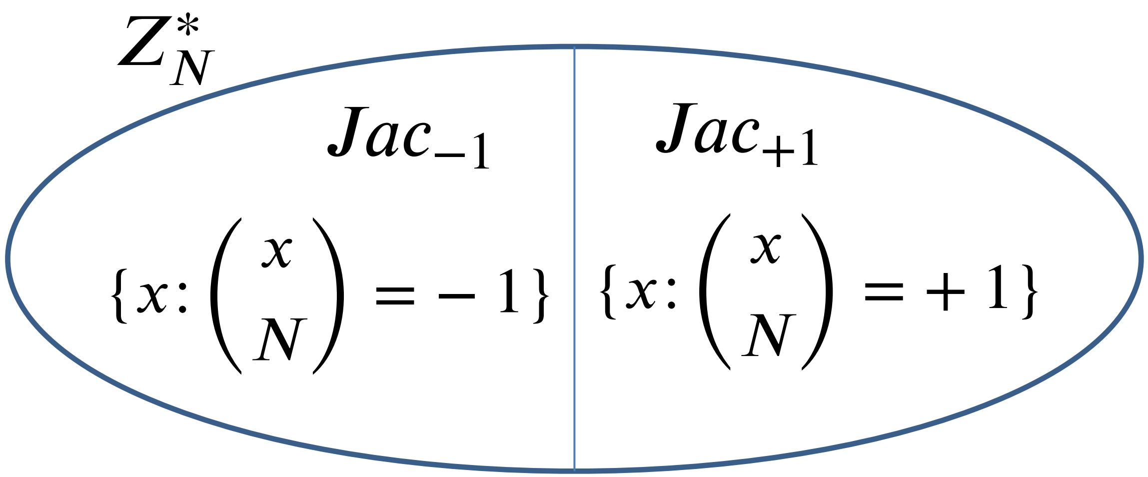
Quadratic Residuosity

Fact: For any odd N, Jac divides Z_N^* evenly *unless N is a perfect square*. (If N is a perfect square, all of Z_N^* has Jacobi symbol +1.)



Quadratic Residuosity

Surprising fact: For any N, Jacobi symbol $\left(\frac{x}{N}\right)$ is computable in poly time **without knowing the prime factorization of N**.

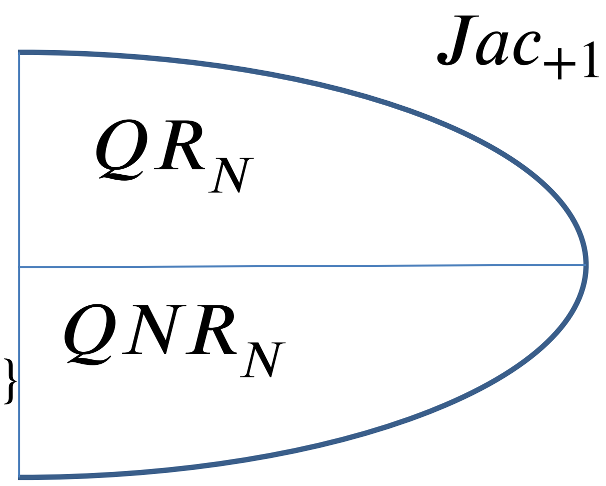


Quadratic Residuosity

Let $N = pq$ be a product of two large primes.

So: $QR_N = \{x : \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = +1\}$

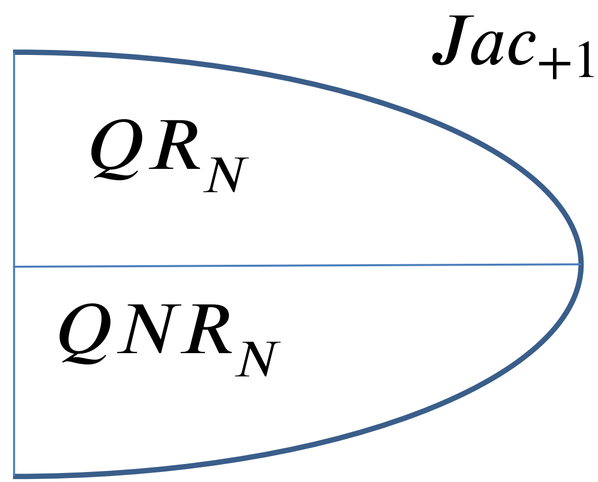
$QNR_N = \{x : \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1\}$



QR_N is the set of squares mod N and QNR_N is the set of non-squares mod N with Jacobi symbol +1.

Quadratic Residuosity

Call an odd integer N good if exactly half the elements of Z_N^* have Jacobi symbol +1, and exactly half of them are squares.

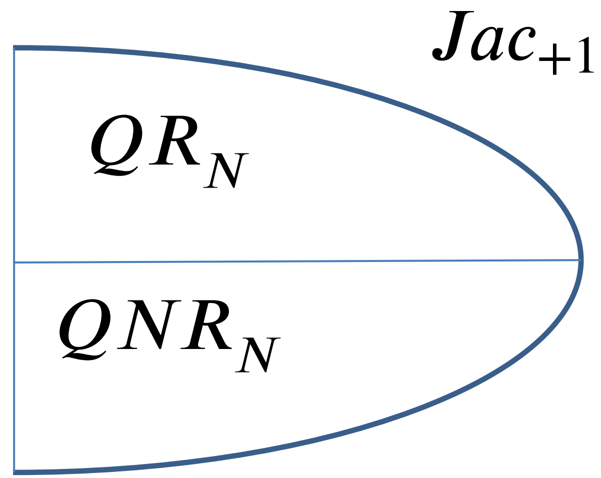


QR_N is the set of squares mod N and QNR_N is the set of non-squares mod N with Jacobi symbol +1.

Quadratic Residuosity

Fact: N is good iff

$$N = p^i q^j \; is\ odd,\ \ and\ i, j \geq 1, \text{ not both even}.$$

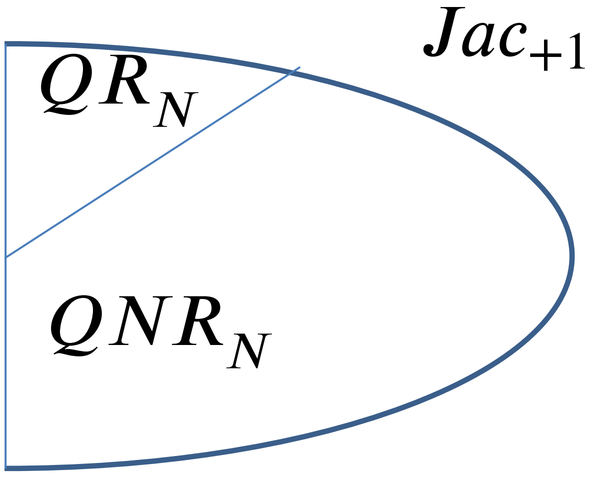


<u>IMPORTANT PROPERTY</u>: If $\underline{y_1}$ and $\underline{y_2}$ are both in $\underline{QNR}$, then their product $\underline{y_1 y_2}$ is in $\underline{QR}$.

Quadratic Residuosity

The fraction of residues smaller if

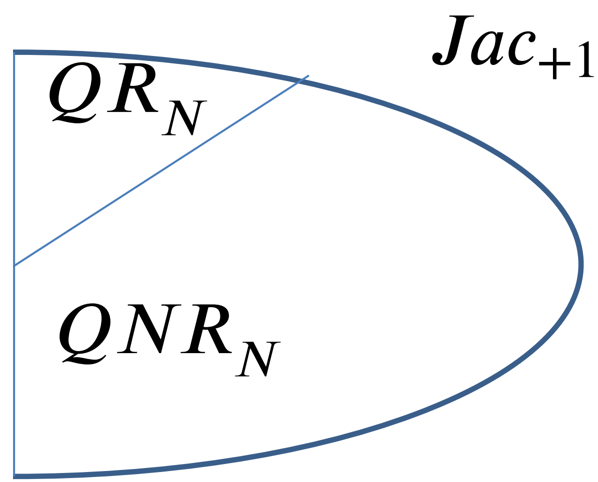
***N* has three or more prime factors!**



Quadratic Residuosity

The fraction of residues smaller if

N has three or more prime factors!



~~**IMPORTANT PROPERTY**: If y_1 and y_2 are both in QNR, then their product $y_1 y_2$ is in QR.~~

Quadratic Residuosity

Let $N = pq$ be a product of two large primes.

<u>Quadratic Residuosity Assumption (QRA)</u>

No PPT algorithm can distinguish between a random element of QR_N from a random element of QNR_N given only N.

HOW TO CONSTRUCT NIZK IN THE CRS MODEL

Step 1. **Review** our number theory hammers & polish them.

Step 2. **Construct** NIZK for a special NP language, namely quadratic *non*-residuosity.

Step 3. **Bootstrap** to NIZK for 3SAT, an NP-complete language.

NIZK for Quadratic Non-Residuosity

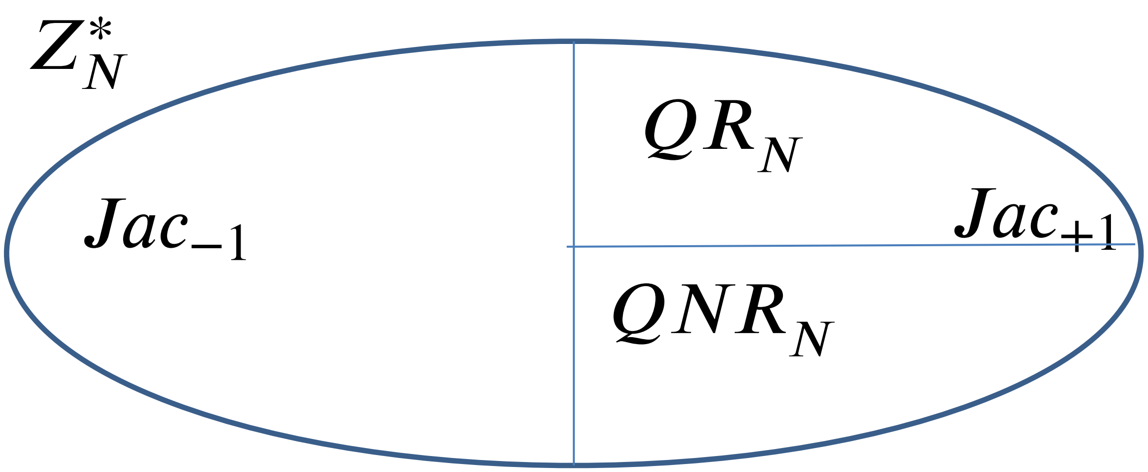
Define the NP language $GOOD$ with instances $(\boldsymbol{N}, \boldsymbol{y})$ where

- N is good; and
- $y \in QNR_N$ (that is, y has Jacobi symbol +1 but is not a square mod N)

NIZK for Quadratic Non-Residuosity

Define the NP language $GOOD$ with instances $(\boldsymbol{N}, \boldsymbol{y})$ where

- N is good; and
- $y \in QNR_N$ (that is, y has Jacobi symbol +1 but is not a square mod N)



NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

$$\mathbf{P} \longrightarrow \mathbf{V}$$

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) $\qquad\qquad\qquad\qquad$ (N, y)

P $\longrightarrow$ **V**

Check:

- N is odd
- N is not a prime power,
- N is not a perfect square;

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

P → **V**

Check:

- N is odd
- N is not a prime power,
- N is not a perfect square;

Fact: If all these pass, then at most half of Jac_N^{+1} are squares.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

P $\longrightarrow$ **V**

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) $\qquad$ (N, y)

$\mathbf{P} \longrightarrow \mathbf{V}$

If N is good and $y \in QNR_N$:

either r_i is in QR_N or yr_i is in QR_N

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

$\mathbf{P}$ $\longrightarrow$ $\mathbf{V}$

If N is good and $y \in QNR_N$:

either r_i is in QR_N or yr_i is in QR_N
so I can compute $\sqrt{r_i}$ or $\sqrt{yr_i}$.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

$$\mathbf{P} \longrightarrow \mathbf{V}$$

If N is good and $y \in QNR_N$:

either r_i is in QR_N or yr_i is in QR_N
so I can compute $\sqrt{r_i}$ or $\sqrt{yr_i}$.

If not … I'll be stuck!

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) **P**

$\forall i: \sqrt{r_i}$ OR $\sqrt{yr_i}$ $\longrightarrow$

(N, y) **V**

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) **P** $\xrightarrow{\forall i:\ \sqrt{r_i}\ \text{OR}\ \sqrt{yr_i}}$ **V** (N, y)

Check:

- N is odd
- N is not a prime power,
- N is not a perfect square; and
- I received either a mod-N square root of r_i or yr_i

NIZK for Quadratic Non-Residuosity

$$\boxed{CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m}$$

(N, y) **P** $\xrightarrow{\forall i:\ \sqrt{r_i} \text{ OR } \sqrt{yr_i}}$ **V** (N, y)

Soundness (what if N has more than 2 prime factors)

No matter what y is, for half the r_i, both r_i and yr_i are ***not*** quadratic residues.

NIZK for Quadratic Non-Residuosity

$$\boxed{CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m}$$

(N, y) (N, y)

$$\mathbf{P} \xrightarrow{\forall i:\ \sqrt{r_i}\ \text{OR}\ \sqrt{yr_i}} \mathbf{V}$$

Soundness (what if N has more than 2 prime factors)

No matter what y is, ***for half the*** r_i, both r_i and yr_i are ***not*** quadratic residues.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) $\quad$ (N, y)

$$\mathbf{P} \xrightarrow{\forall i: \sqrt{r_i} \text{ OR } \sqrt{yr_i}} \mathbf{V}$$

Soundness (what if y is a residue)

Then, if r_i happens to be a non-residue, both r_i and yr_i are ***not*** quadratic residues.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) $\mathbf{P}$ $\xrightarrow{\forall i : \pi_i = \sqrt{r_i} \text{ OR } \sqrt{yr_i}}$ $\mathbf{V}$ (N, y)

(Perfect) Zero Knowledge Simulator S:

First pick the proof π_i to be random in Z_N^*.

Then, *reverse-engineer* the CRS, letting $r_i = \pi_i^2$ or $r_i = \pi_i^2 / y$ randomly.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) $\qquad$ (N, y)

$$\mathbf{P} \xrightarrow{\forall i: \pi_i = \sqrt{r_i} \text{ OR } \sqrt{y r_i}} \mathbf{V}$$

(Perfect) Zero Knowledge Simulator S:

First pick the proof π_i to be random in Z_N^*.

Then, *reverse-engineer* the CRS, letting $r_i = \pi_i^2$ or $r_i = \pi_i^2 / y$ randomly.

NIZK for Quadratic Non-Residuosity

$$\boxed{CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m}$$

(N, y) $\qquad\qquad\qquad\qquad$ (N, y)

$$\mathbf{P} \longrightarrow \mathbf{V}$$

CRS depends on the instance N. Not good.

NIZK for Quadratic Non-Residuosity

$$CRS = (r_1, r_2, \ldots, r_m) \leftarrow (Jac_N^{+1})^m$$

(N, y) (N, y)

P $\longrightarrow$ **V**

CRS depends on the instance N. Not good.

Soln: Let CRS be random numbers.

Interpret them as elements of Z_N^* and both the prover and verifier filter out Jac_N^{-1}.

NEXT LECTURE

Step 1. **Review** our number theory hammers & polish them.

Step 2. **Construct** NIZK for a special NP language, namely quadratic *non*-residuosity.

Step 3. **Bootstrap** to NIZK for 3SAT, an NP-complete language.

3SAT

Boolean Variables: x_i can be either true (1) or false (0)

A Literal is either x_i or $\bar{x}_i$.

3SAT

Boolean Variables: x_i can be either true (1) or false (0)

A Literal is either x_i or $\bar{x}_i$.

A Clause is a *disjunction* of literals.

E.g. $x_1 \lor x_2 \lor \bar{x}_5$

3SAT

<u>Boolean Variables</u>: x_i can be either true (1) or false (0)

A <u>Literal</u> is either x_i or $\bar{x}_i$.

A <u>Clause</u> is a *disjunction* of literals.

E.g. $x_1 \vee x_2 \vee \bar{x}_5$

A <u>Clause</u> is true if any one of the literals is true.

3SAT

Boolean Variables: x_i can be either true (1) or false (0)

A Literal is either x_i or $\bar{x}_i$.

A Clause is a *disjunction* of literals.

E.g. $x_1 \vee x_2 \vee \bar{x}_5$ is true as long as:

$$(x_1, x_2, x_5) \neq (0,0,1)$$

3SAT

<u>Boolean Variables</u>: x_i can be either true (1) or false (0)

A <u>Literal</u> is either x_i or $\bar{x}_i$.

A <u>3-Clause</u> is a *disjunction* of 3-literals.

3SAT

Boolean Variables: x_i can be either true (1) or false (0)

A Literal is either x_i or $\bar{x}_i$.

A 3-Clause is a *disjunction* of 3-literals.

A 3-SAT formula is a *conjunction* of many 3-clauses.

E.g. $\boldsymbol{\Psi} = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_1 \vee x_3 \vee x_4) (\bar{x}_2 \vee x_3 \vee \bar{x}_5)$

3SAT

Boolean Variables: x_i can be either true (1) or false (0)

A Literal is either x_i or $\bar{x}_i$.

A 3-Clause is a *disjunction* of 3-literals.

A 3-SAT formula is a *conjunction* of many 3-clauses.

E.g. $\mathbf{\Psi} = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_1 \vee x_3 \vee x_4) (\bar{x}_2 \vee x_3 \vee \bar{x}_5)$

A 3-SAT formula $\mathbf{\Psi}$ is **satisfiable** if there is an assignment of values to the variables x_i that makes all its clauses true.

3SAT

Cook-Levin Theorem: It is NP-complete to decide whether a <u>3-SAT formula</u> $\mathbf{\Psi}$ is satisfiable.

A <u>3-SAT formula</u> is a *conjunction* of many 3-clauses.

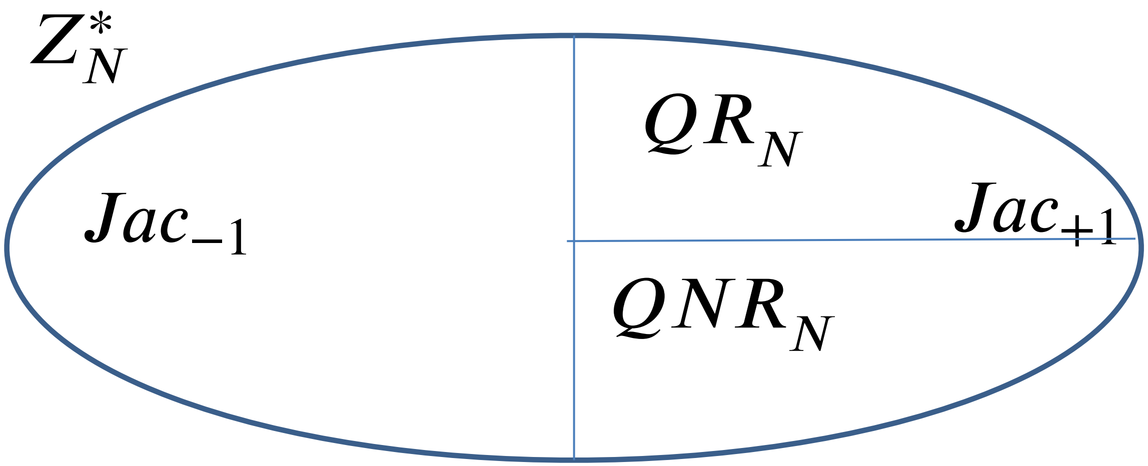
E.g. $\mathbf{\Psi} = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_1 \vee x_3 \vee x_4) (\bar{x}_2 \vee x_3 \vee \bar{x}_5)$

A <u>3-SAT formula</u> $\mathbf{\Psi}$ is **satisfiable** if there is an assignment of values to the variables x_i that makes all its clauses true.

NIZK for 3SAT: Recall…

We saw a way to show that a pair $(\boldsymbol{N}, \boldsymbol{y})$ is GOOD. That is:

- the following is the picture of Z_N^* and
- for every $r \in Jac_{+1}$, either r or ry is a quadratic residue.



NIZK for 3SAT

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** **V**

Input: $\Psi = (x_1 \vee\ x_2 \vee \bar{x}_5) \wedge\ (x_1 \vee\ x_3 \vee x_4)\ (\bar{x}_2 \vee\ x_3 \vee \bar{x}_5)$

n variables, m clauses.

NIZK for 3SAT

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** $\xrightarrow{(N, y, \pi)}$ **V**

1. Prover picks an (N, y) and proves that it is GOOD.

Input: $\Psi = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_1 \vee x_3 \vee x_4) (\bar{x}_2 \vee x_3 \vee \bar{x}_5)$

n variables, m clauses.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** $\xrightarrow{(N, y, \pi)}$ **V**

1. Prover picks an (N, y) and proves that it is GOOD.

Input: $\Psi = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_1 \vee x_3 \vee x_4)\ (\bar{x}_2 \vee x_3 \vee \bar{x}_5)$

n variables, m clauses.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** $\xrightarrow{(N, y, \pi)}$ **V**

2. Prover encodes the satisfying assignment

$y_i \leftarrow QR_N$ if x_i is false

$y_i \leftarrow QNR_N$ if x_i is true

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ $\mathbf{P}$ $\xrightarrow{(N, y, \pi)}$ $\mathbf{V}$

2. Prover encodes the satisfying assignment & ∴ the literals

$Enc(x_i) = y_i$, then $Enc(\bar{x}_i) = yy_i$

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** $\xrightarrow{(N, y, \pi)}$ **V**

2. Prover encodes the satisfying assignment & $\therefore$ the literals

$Enc(x_i) = y_i$, then $Enc(\bar{x}_i) = yy_i$

$\therefore$ exactly one of $Enc(x_i)$ or $Enc(\bar{x}_i)$ is a non-residue.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** Encode vars: $(y_1, \ldots, y_n)$ $\rightarrow$ **V**

2. Prover encodes the satisfying assignment & ∴ the literals

$Enc(x_i) = y_i$, then $Enc(\bar{x}_i) = yy_i$

∴ exactly one of $Enc(x_i)$ or $Enc(\bar{x}_i)$ is a non-residue.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** Encode vars: $(y_1, \ldots, y_n)$ $\rightarrow$ **V**

3. Prove that (encoded) assignment satisfies each clause.

For each clause, say $x_1 \vee x_2 \vee \bar{x}_5$,

let (a_1, b_1, c_1) denote the encoded variables.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_n)$ **P** Encode vars: $(y_1, \ldots, y_n)$ $\longrightarrow$ **V**

3. Prove that (encoded) assignment satisfies each clause.

For each clause, say $x_1 \vee \ x_2 \vee \bar{x}_5$,

let (a_1, b_1, c_1) denote the encoded variables.

So, each of them is either y_i (if the literal is a var) or yy_i (if the literal is a negated var).

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** Encode vars: $(y_1, \ldots, y_n)$ **V**

3. Prove that (encoded) assignment satisfies each clause.

For each clause, say $x_1 \vee \; x_2 \vee \bar{x}_5$, let

$(a_1 = y_1, b_1 = y_2, c_1 = yy_5)$ denote the encoded variables

So, each of them is either y_i (if the literal is a var) or yy_i

(if the literal is a negated var).

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ $\quad$ (N, y, π) $\quad$ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ $\quad$ $\mathbf{P}$ $\quad$ Encode vars: $(y_1, \ldots, y_n)$ $\longrightarrow$ $\mathbf{V}$

3. Prove that (encoded) assignment satisfies each clause.

For each clause, say $x_1 \vee \ x_2 \vee \bar{x}_5$,

let (a_1, b_1, c_1) denote the encoded variables.

WANT to SHOW: $x_1\ OR\ x_2\ OR\ \bar{x}_5$ is true.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_\mathrm{n})$ $\mathbf{P}$ Encode vars: $(y_1, \ldots, y_n)$ $\longrightarrow$ $\mathbf{V}$

3. Prove that (encoded) assignment satisfies each clause.

For each clause, say $x_1 \vee x_2 \vee \bar{x}_5$,

let (a_1, b_1, c_1) denote the encoded variables.

WANT to SHOW: $a_1\ OR\ b_1\ OR\ c_1$ is a non-residue.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ $\quad\quad (N, y, \pi) \quad\quad$ Ψ

Satisfying assignment $(\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_n)$ **P** $\xrightarrow{\text{Encode vars: } (y_1, \ldots, y_n)}$ **V**

3. Prove that (encoded) assignment satisfies each clause?

For each clause, say $x_1 \vee\ x_2 \vee \bar{x}_5$,

let (a_1, b_1, c_1) denote the encoded variables.

WANT to SHOW: $a_1\ OR\ b_1\ OR\ c_1$ is a non-residue.

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1 \; OR \; b_1 \; OR \; c_1$ is a non-residue.

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1 \; OR \; b_1 \; OR \; c_1$ is a non-residue.

Equiv: The “pattern” of $(a_1, \; b_1, c_1)$ is **NOT** (QR, QR, QR).

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1 \; OR \; b_1 \; OR \; c_1$ is a non-residue.

Equiv: The “pattern” of $(a_1, \; b_1, c_1)$ is **NOT** (QR, QR, QR).

CLEVER IDEA: Generate seven *additional* triples

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1\ OR\ b_1\ OR\ c_1$ is a non-residue.

Equiv: The "pattern" of $(a_1,\ b_1, c_1)$ is **NOT** (QR, QR, QR).

CLEVER IDEA: Generate seven *additional* triples

$$(a_1,\ b_1, c_1)$$
$$(a_2,\ b_2, c_2)$$
$$(a_3,\ b_3, c_3)$$
$$(a_4,\ b_4, c_4)$$
$$(a_5,\ b_5, c_5)$$
$$(a_6,\ b_6, c_6)$$
$$(a_7,\ b_7, c_7)$$
$$(a_8,\ b_8, c_8)$$

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1 \ OR \ b_1 \ OR \ c_1$ is a non-residue.

Equiv: The "pattern" of $(a_1, \ b_1, c_1)$ is **NOT** (QR, QR, QR).

CLEVER IDEA: Generate seven *additional* triples

original triple $\rightarrow$ $(a_1, \ b_1, c_1)$

$(a_2, \ b_2, c_2)$

$(a_3, \ b_3, c_3)$

$(a_4, \ b_4, c_4)$

$(a_5, \ b_5, c_5)$

$(a_6, \ b_6, c_6)$

$(a_7, \ b_7, c_7)$

$(a_8, \ b_8, c_8)$

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1\ OR\ b_1\ OR\ c_1$ is a non-residue.

Equiv: The "pattern" of $(a_1,\ b_1, c_1)$ is **NOT** (QR, QR, QR).

CLEVER IDEA: Generate seven *additional* triples

original triple $(a_1,\ b_1, c_1)$

show this is a QR: reveal the square roots $(a_2,\ b_2, c_2)$

$(a_3,\ b_3, c_3)$

$(a_4,\ b_4, c_4)$

$(a_5,\ b_5, c_5)$

$(a_6,\ b_6, c_6)$

$(a_7,\ b_7, c_7)$

$(a_8,\ b_8, c_8)$

NIZK for 3SAT

Prove that (encoded) assignment satisfies each clause.

WANT to SHOW: $a_1 \; OR \; b_1 \; OR \; c_1$ is a non-residue.

Equiv: The "pattern" of $(a_1, \; b_1, c_1)$ is **NOT** (QR, QR, QR).

CLEVER IDEA: Generate seven *additional* triples

original triple

show this is a QR:
reveal the square roots

$(a_1, \; b_1, c_1)$
$(a_2, \; b_2, c_2)$
$(a_3, \; b_3, c_3)$
$(a_4, \; b_4, c_4)$
$(a_5, \; b_5, c_5)$
$(a_6, \; b_6, c_6)$
$(a_7, \; b_7, c_7)$
$(a_8, \; b_8, c_8)$

"Proof of Coverage":
show that the 8 triples span
all possible QR patterns

NIZK for 3SAT

CLEVER IDEA: Generate seven *additional* triples

original triple

show this is a QR: reveal the square roots

$(a_1,\ b_1, c_1)$
$(a_2,\ b_2, c_2)$
$(a_3,\ b_3, c_3)$
$(a_4,\ b_4, c_4)$
$(a_5,\ b_5, c_5)$
$(a_6,\ b_6, c_6)$
$(a_7,\ b_7, c_7)$
$(a_8,\ b_8, c_8)$

"Proof of Coverage": show that the 8 triples span all possible QR signatures

<u>Proof of Coverage</u>: For each of poly many triples $\underline{(r, s, t)}$ from CRS, show one of the 8 triples has the same signature.

That is, there is a triple (a_i, b_i, c_i) s.t. (ra_i, sb_i, tc_i) is (QR, QR, QR).

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ $\quad (N, y, \pi) \quad$ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ $\quad$ **P** $\quad$ **V**

Encode vars: $(y_1, \ldots, y_n)$

For each clause ψ: ρ_ψ

3. Prove that (encoded) assignment satisfies each clause.

For each clause, construct the proof ρ = (7 additional triples, square root of the second triples, proof of coverage).

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ $\quad\quad (N, y, \pi) \quad\quad$ Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P** $\quad$ **V**

Encode vars: $(y_1, \ldots, y_n)$

For each clause ψ: ρ_ψ

Completeness & Soundness: Exercise.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

	Ψ	(N, y, π)	Ψ
Satisfying assignment $(w_1, w_2, \ldots, w_n)$	$\mathbf{P}$	Encode vars: $(y_1, \ldots, y_n)$ → For each clause ψ: ρ_ψ	$\mathbf{V}$

Completeness & Soundness: Exercise.

Zero Knowledge: Simulator picks (N, y) where y is a quadratic **residue**.

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ (N, y, π) Ψ

Satisfying assignment $(w_1, w_2, \ldots, w_n)$ **P**

Encode vars: $(y_1, \ldots, y_n)$

For each clause ψ: ρ_ψ

V

Completeness & Soundness: Exercise.

Zero Knowledge: Simulator picks (N, y) where y is a quadratic **residue**.

Now, encodings of ALL the literals can be set to TRUE!!

NIZK for 3SAT

$$CRS = (r_1, r_2, \ldots, r_{large\ number}) \leftarrow (Jac_N^{+1})^{large\ number}$$

Ψ $\qquad$ (N, y, π) $\qquad$ Ψ

Satisfying assignment $(\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_\mathrm{n})$ $\quad$ **P** $\quad$ Encode vars: $(y_1, \ldots, y_n)$ $\longrightarrow$ **V**

For each clause ψ: ρ_ψ

Completeness & Soundness: Exercise.

Zero Knowledge: Simulator picks (N, y) where y is a quadratic **residue**.

Now, encodings of ALL the literals can be set to TRUE!!

An Application of NIZK:

Non-malleable and Chosen Ciphertext Secure Encryption Schemes

Non-Malleability

$m \leftarrow \mathrm{Dec}(\mathbf{sk}, c)$

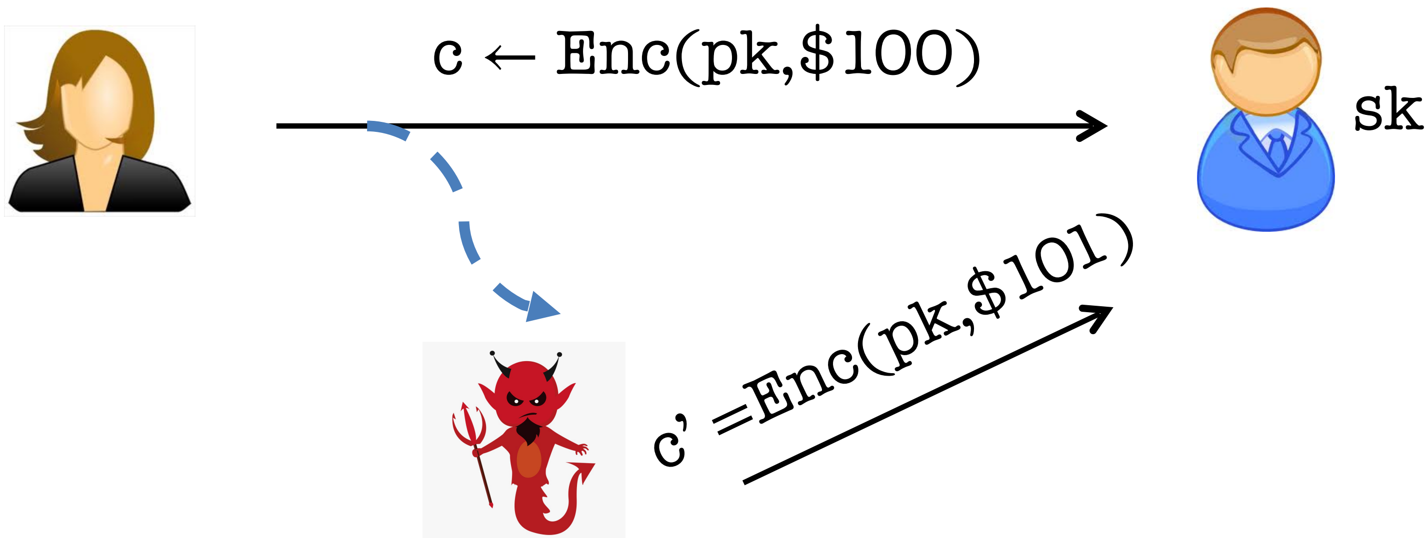
$c \leftarrow \mathrm{Enc}(\mathbf{pk}, m)$

sk

Public-key directory

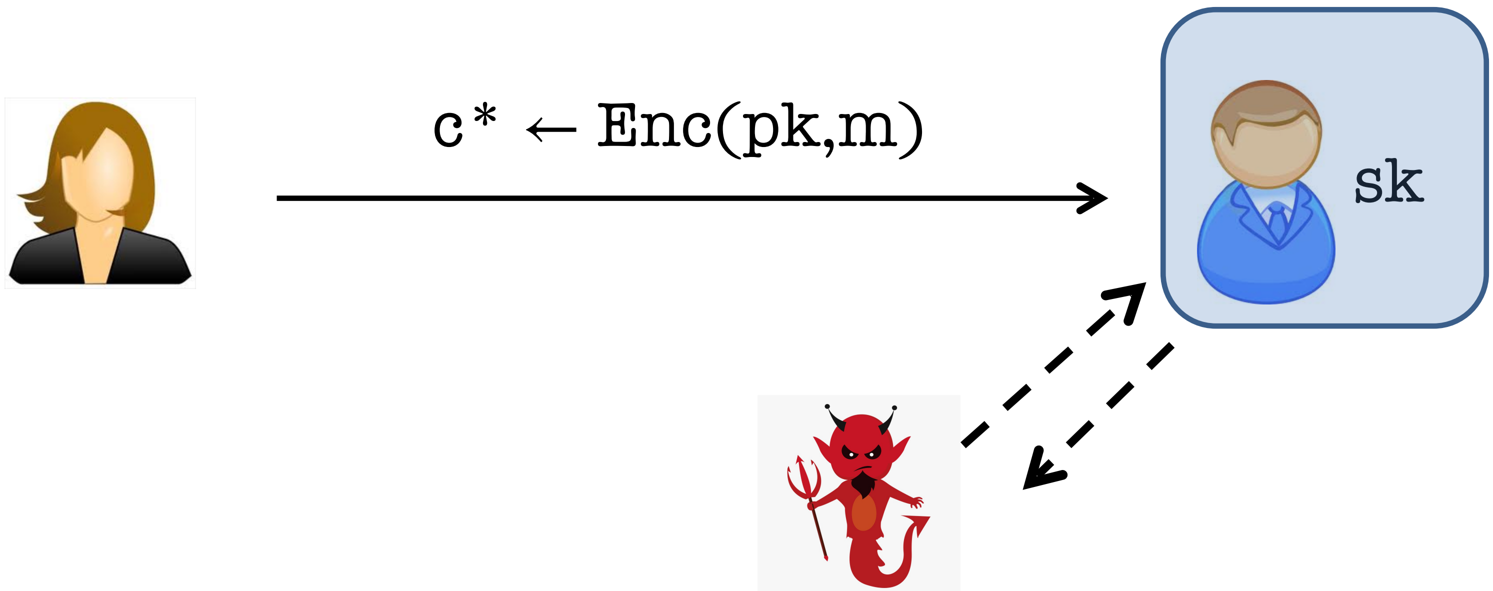
Bob	**pk**

Active Attacks 1: Malleability



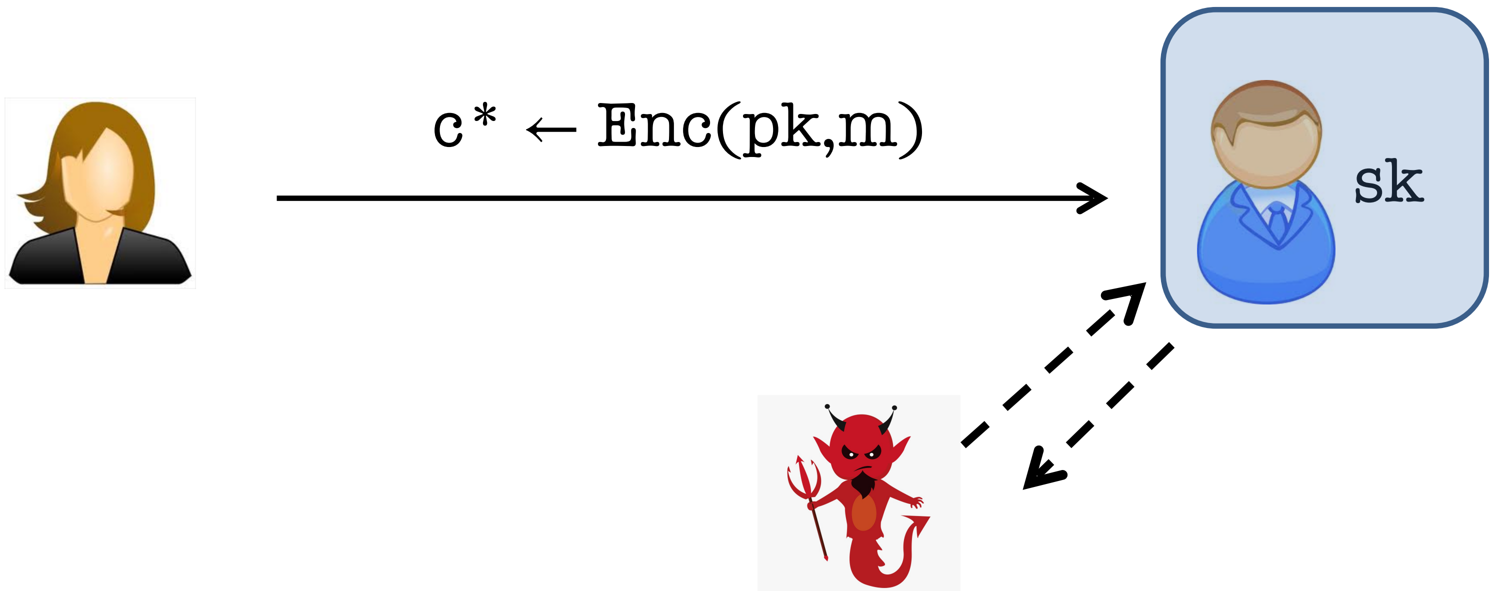
ATTACK: Adversary could modify ("maul") an encryption of m into an encryption of a related message m'.

Active Attacks 2: Chosen-Ciphertext Attack



ATTACK: Adversary may have access to a decryption "oracle" and can use it to break security of a "target" ciphertext c^* or even extract the secret key!

Active Attacks 2: Chosen-Ciphertext Attack



In fact, Bleichenbacher showed how to extract the entire secret key given only a “ciphertext verification” oracle.

Challenger

IND-CCA Security

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$

$m_0^*, m_1^* \;\; s.t. \; |m_0^*| = |m_1^*|$

$b \leftarrow \{0,1\};\; c^* \leftarrow Enc(pk, m_b^*)$

$\boldsymbol{c}^*$

b'

Eve wins if $b' = b$.
IND-CCA secure if no PPT Eve can win with prob. > $\frac{1}{2} + \mathrm{negl}(n)$.

IND-CCA Security

Challenger

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$

$m_0^*, m_1^* \ \ s.t. \ \ |m_0^*| = |m_1^*|$

$b \leftarrow \{0,1\};\ c^* \leftarrow Enc(pk, m_b^*)$

$\boldsymbol{c^*}$

$\boldsymbol{c_i}$

$\boldsymbol{Dec(sk, c_i)}$

b'

Eve wins if $b' = b$.
IND-CCA secure if no PPT Eve can win with prob. > $\frac{1}{2} + \mathrm{negl}(n)$.

IND-CCA Security

Challenger

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$ $\longrightarrow$

m_0^*, m_1^* $s.t.$ $|m_0^*| = |m_1^*|$

$\longleftarrow$

$b \leftarrow \{0,1\}$; $c^* \leftarrow Enc(pk, m_b^*)$

c^*

$\longrightarrow$

$c_i \neq c^*$

$\longleftarrow$

$Dec(sk, c_i)$

$\longrightarrow$

b'

$\longleftarrow$

Eve wins if $b' = b$.

IND-CCA secure if no PPT Eve can win with prob. > $\frac{1}{2} + \mathrm{negl}(n)$.

Challenger

IND-CCA Security

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$

$m_0^*, m_1^* \;\; s.t. \;\; |m_0^*| = |m_1^*|$

$b \leftarrow \{0,1\};\ c^* \leftarrow Enc(pk, m_b^*)$

$\boldsymbol{c^*}$

$\boldsymbol{c_i \neq c^*}$

$\boldsymbol{Dec(sk, c_i)}$

b'

Eve wins if $b' = b$.
IND-CCA secure if no PPT Eve can win with prob. > $\frac{1}{2} + \text{negl}(n)$.

IND-CCA Security

Challenger

Eve

pk

$(pk, sk) \leftarrow Gen(1^n)$

c_i

$Dec(sk, c_i)$

$m_0^*, m_1^* \;\; s.t. \;\; |m_0^*| = |m_1^*|$

$b \leftarrow \{0,1\}; \; c^* \leftarrow Enc(pk, m_b^*)$

c^*

$c_i \neq c^*$

$Dec(sk, c_i)$

b'

Eve wins if $b' = b$.

IND-CCA secure if no PPT Eve can win with prob. > $\frac{1}{2} + \mathrm{negl}(n)$.

Constructing CCA-Secure Encryption
(Intuition)

NIZK Proofs of Knowledge should help!

Constructing CCA-Secure Encryption (Intuition)

NIZK Proofs of Knowledge should help!

Idea: The encrypting party attaches an NIZK proof of knowledge of the underlying message to the ciphertext.

$$C\colon\ (\mathrm{c} = \mathrm{CPAEnc}(m; r), \mathrm{proof}\ \pi\ \textit{that “I know m and r”})$$

Constructing CCA-Secure Encryption (Intuition)

NIZK Proofs of Knowledge should help!

Idea: The encrypting party attaches an NIZK proof of knowledge of the underlying message to the ciphertext.

$$C\colon\ (\mathrm{c} = \mathrm{CPAEnc}(m; r), \mathrm{proof}\ \pi\ \textit{that "I know m and r"})$$

This idea will turn out to be useful, but NIZK proofs themselves can be malleable!

Constructing CCA-Secure Encryption
(Intuition)

OUR GOAL: Hard to modify an encryption of m into an encryption of a related message, say m+1.

Constructing CCA-Secure Encryption
(Intuition)

OUR GOAL: **Hard to modify** an encryption of m into an encryption of a related message, say m+1.

Constructing CCA-Secure Encryption
(Intuition)

Digital Signatures should help!

OUR GOAL: Hard to modify an encryption of m into an encryption of a related message, say m+1.

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$= \mathrm{CPAEnc}\left(pk, m; r\right), Sign(c))$$

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$(\mathrm{c} = \mathrm{CPAEnc}\left(pk, m; r\right), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $\left(sgk, vk\right) \leftarrow Sign.Gen(1^n)$

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$(\mathrm{c} = \mathrm{CPAEnc}\left(pk, m; r\right), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $\left(sgk, vk\right) \leftarrow Sign.Gen(1^n)$

Is this CCA-secure/non-malleable?

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$(\mathrm{c} = \mathrm{CPAEnc}\left(pk, m; r\right), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $\left(sgk, vk\right) \leftarrow Sign.Gen(1^n)$

Is this CCA-secure/non-malleable?

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$(\mathrm{c} = \mathrm{CPAEnc}\left(pk, m; r\right), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $\left(sgk, vk\right) \leftarrow Sign.Gen(1^n)$

Is this CCA-secure/non-malleable?

If the adversary changes vk, all bets are off!

Constructing CCA-Secure Encryption

Let's start with Digital Signatures.

$$(\mathrm{c} = \mathrm{CPAEnc}\left(pk, m; r\right), Sign_{sgk}(c), vk)$$

where the encryptor produces a signing / verification key pair by running $\left(sgk, vk\right) \leftarrow Sign.Gen(1^n)$

Is this CCA-secure/non-malleable?

If the adversary changes vk, all bets are off!

Lesson: NEED to "tie" the ciphertext c to vk in a "meaningful" way.

Observation:
IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}(pk, m; r)$,
can an adversary produce $\mathrm{CPAEnc}(pk', m+1; r)$?

NO! Suppose she could. Then, I can come up with a reduction that breaks the IND-CPA security of $\mathrm{CPAEnc}(pk, m; r)$.

Observation:

IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}\left(pk, m; r\right)$,
can an adversary produce $\mathrm{CPAEnc}\left(pk', m+1; r\right)$?

pk, pk' $\longrightarrow$

$CPAEnc(pk, m)$ $\longrightarrow$

$\boldsymbol{CPAEnc(pk', m+1)}$ $\longleftarrow$

Diff-Key NM adversary

Observation:

IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}(pk, m; r)$,
can an adversary produce $\mathrm{CPAEnc}(pk', m+1; r)$?

Reduction = CPA adversary

pk, pk'

$\longrightarrow$

$CPAEnc(pk, m)$

$\longrightarrow$

$\boldsymbol{CPAEnc(pk', m+1)}$

$\longleftarrow$

Diff-Key NM adversary

Observation:

IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}\left(pk, m; r\right)$,
can an adversary produce $\mathrm{CPAEnc}\left(pk', m+1; r\right)$?

Reduction = CPA adversary

pk

$CPAEnc(pk, m)$

pk, pk'

$CPAEnc(pk, m)$

$CPAEnc(pk', m+1)$

Diff-Key NM adversary

Observation:

IND-CPA $\Longrightarrow$ "Different-Key Non-malleability"

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}\left(pk, m; r\right)$,

can an adversary produce $\mathrm{CPAEnc}\left(pk', m+1; r\right)$?

Reduction = CPA adversary

Pick (pk', sk')

pk

$CPAEnc(pk, m)$

pk, pk'

$CPAEnc(pk, m)$

$CPAEnc(pk', m+1)$

Diff-Key NM adversary

Observation:

IND-CPA $\Longrightarrow$ “Different-Key Non-malleability”

Different-Key NM: Given $pk, pk', \mathrm{CPAEnc}\left(pk, m; r\right)$,
can an adversary produce $\mathrm{CPAEnc}\left(pk', m+1; r\right)$?

Reduction = CPA adversary

$\xrightarrow{pk}$

$\xrightarrow{CPAEnc(pk,m)}$

$\xleftarrow{m}$

Pick (pk', sk')

$\xrightarrow{pk,\, pk'}$

$\xrightarrow{CPAEnc(pk,m)}$

$\xleftarrow{CPAEnc(pk', m+1)}$

Decrypt and subtract 1.

Diff-Key NM adversary

Putting it together

CCA Public Key: $2n$ public keys of the CPA scheme

(where $n = |vk|$)

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \dots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

Putting it together

CCA Public Key: $2n$ **public keys of the CPA scheme**

(where $n = |vk|$)

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

Putting it together

CCA Public Key: $2n$ **public keys of the CPA scheme**

(where $n = |vk|$)

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Putting it together

CCA Public Key: $2n$ public keys of the CPA scheme

(where $n = |vk|$)

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

Putting it together

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \\ & & & \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

Putting it together

Non-malleability rationale: Either

- Adversary keeps vk the same (in which case she has to break the signature scheme); or
- She changes the vk in which case she breaks the diff-NM game, and therefore CPA security.

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

Call it a day?

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \\ & & & \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

Call it a day?

We are not done!! Adversary could create ill-formed ciphertexts (e.g. the different cts encrypt different messages) and uses it for a Bleichenbacher-like attack.

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m)$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

NIZK Proofs to the Rescue…

CCA Public Key: $2n$ **public keys of the CPA scheme**

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

NIZK Proofs to the Rescue…

CCA Public Key: $2n$ public keys of the CPA scheme

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = **NIZK proof that "CT is well-formed"**

Output $(CT,\ vk, \sigma = Sign(sgk, CT))$.

NIZK Proofs to the Rescue…

CCA Public Key: $2n$ public keys of the CPA scheme

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = **NIZK proof that "CT is well-formed"**

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

NIZK Proofs to the Rescue…

CCA Public Key: $2n$ public keys of the CPA scheme

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}, \quad \color{blue}{CRS}$$

NP statement: "there exist $m, r_{i,j}$ such that each $ct_{i,j} = CPAEnc(pk_{i,j}, m; r_{i,j})$"

key pair (sgk, vk)

$$\begin{bmatrix} \cdots k_2 & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = **NIZK proof that "CT is well-formed"**

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

NIZK Proofs to the Rescue...

CCA Public Key: $2n$ public keys of the CPA scheme

$$\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}, \; CRS$$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = NIZK proof that "CT is well-formed"

Output $(CT, \pi, vk, \sigma = Sign(sgk, (CT, \pi)))$.

Are there other attacks?

Did we miss anything else?

Are there other attacks?

Did we miss anything else?

Turns out NO. We can prove that this is CCA-secure.

The Encryption Scheme

CCA Keys:

PK= $\begin{bmatrix} pk_{1,0} & pk_{2,0} & \cdots & pk_{n,0} \\ pk_{1,1} & pk_{2,1} & & pk_{n,1} \end{bmatrix}, CRS$ **SK=** $\begin{bmatrix} sk_{1,0} \\ sk_{1,1} \end{bmatrix}$

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = NIZK proof that "CT is well-formed"

Output $(CT, \pi,\ vk, \sigma = Sign(sgk, (CT, \pi)))$.

The Encryption Scheme

CCA Encryption:

First, pick a sign/ver key pair (sgk, vk)

$$CT = \begin{bmatrix} ct_{1,vk_1} & ct_{2,vk_2} & \cdots & ct_{n,vk_n} \end{bmatrix}$$

where $ct_{i,j} \leftarrow CPAEnc(pk_{i,j}, m; r_{i,j})$

π = NIZK proof that "CT is well-formed"

Output $(CT, \pi, \ vk, \sigma = Sign(sgk, (CT, \pi)))$.

CCA Decryption:

Check the signature.

Check the NIZK proof.

Decrypt with sk_{1,vk_1}.

Proof Sketch

Let's play the CCA game with the adversary.

We will use her to break either the NIZK soundness/ZK, the signature scheme or the CPA-secure scheme.

Proof Sketch

Let’s play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Proof Sketch

Let's play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Hybrid 1: Observe that $\boldsymbol{vk_i} \neq \boldsymbol{vk}^*$.

(Otherwise break signature)

Proof Sketch

Let's play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Hybrid 1: Observe that $\boldsymbol{vk_i} \neq \boldsymbol{vk}^*$.

(Otherwise break signature)

Observe that this means each query ciphertext-tuple involves a different public-key from the challenge ciphertext. Use the "different private-key" to decrypt.

(If the adv sees a difference, she broke NIZK soundness)

Proof Sketch

Let's play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Hybrid 1: Observe that $\boldsymbol{vk_i} \neq \boldsymbol{vk}^*$.

(Otherwise break signature)

Observe that this means each query ciphertext-tuple involves a different public-key from the challenge ciphertext. Use the "different private-key" to decrypt.

(If the adv sees a difference, she broke NIZK soundness)

Hybrid 2: Now change the CRS/π into simulated CRS/π!

(OK by ZK)

Proof Sketch

Let's play the CCA game with the adversary.

Hybrid 0: Play the CCA game as prescribed.

Hybrid 1: Observe that $\boldsymbol{vk_i} \neq \boldsymbol{vk}^*$.

(Otherwise break signature)

Observe that this means each query ciphertext-tuple involves a different public-key from the challenge ciphertext. Use the "different private-key" to decrypt.

(If the adv sees a difference, she broke NIZK soundness)

Hybrid 2: Now change the CRS/π into simulated CRS/π!

(OK by ZK)

If the Adv wins in this hybrid, she breaks **IND-CPA!**