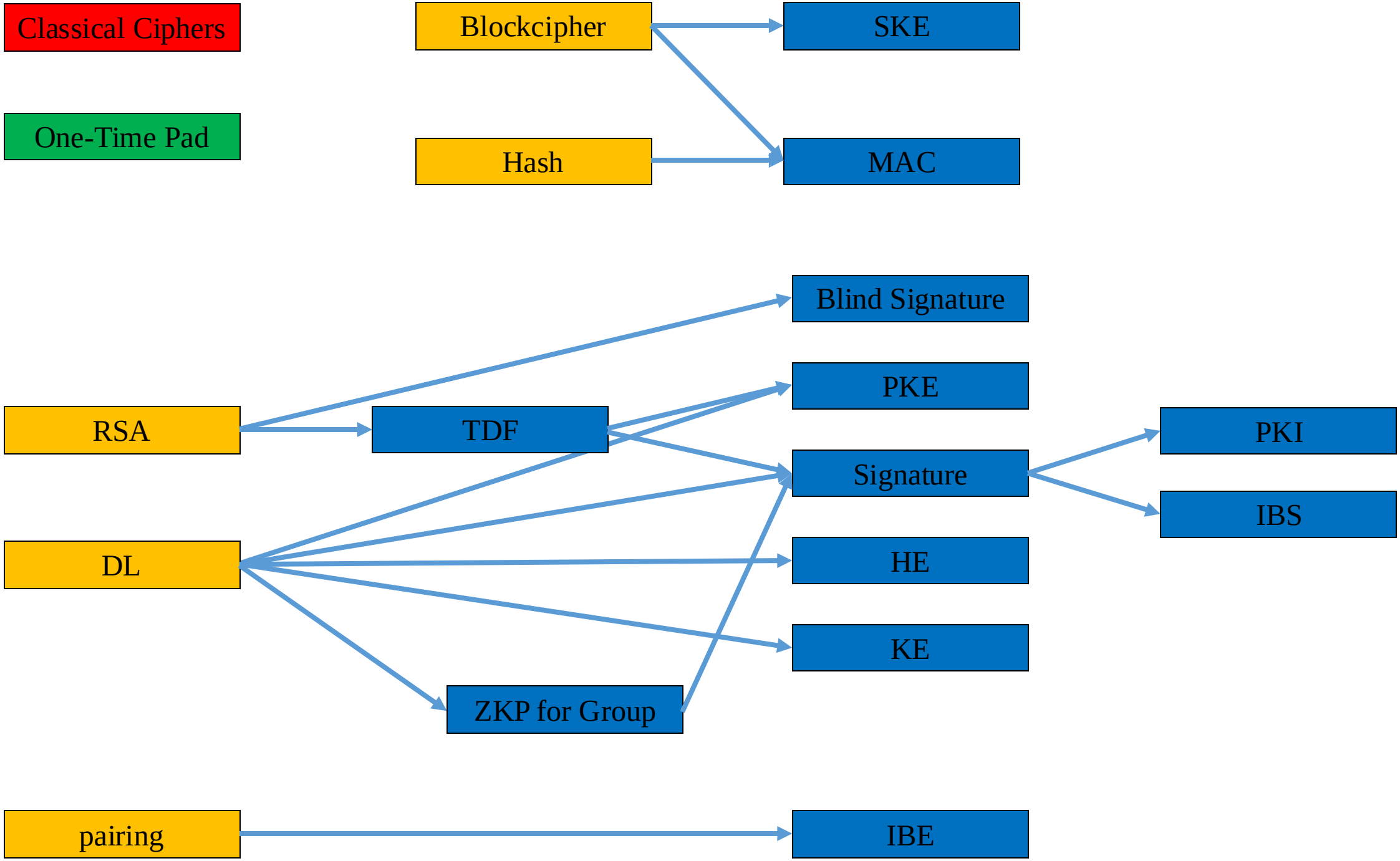
CSCI471/971
Modern Cryptography
Zero-Knowledge Proof II

Rupeng Yang
SCIT UOW

RoadMap

- Week 1-2: Preliminaries
- Week 3-4: Symmetric-Key Cryptography
- Week 5-9: Public-Key Cryptography
- Week 10-11: Zero-Knowledge Proofs


Classical Ciphers
Blockcipher
SKE
One-Time Pad
Hash
MAC
Blind Signature
PKE
RSA
TDF
PKI
Signature
IBS
DL
HE
KE
ZKP for Group
pairing
IBE

Zero-Knowledge Proofs (Revision)

Zero Knowledge Proofs (of knowledge)

- A protocol involving a prover and a verifier.
- The prover takes as input a statement x and a witness w.
- The verifier takes as input the statement x.
- The prover's goal is to convince the verifier that some statement is true (or that she holds the witness w) without revealing any other information.
- Here, we only consider internal attackers, i.e., the prover will try to cheat the verifier and the verifier will try to learn the witness w.

Zero Knowledge Proofs (of knowledge)

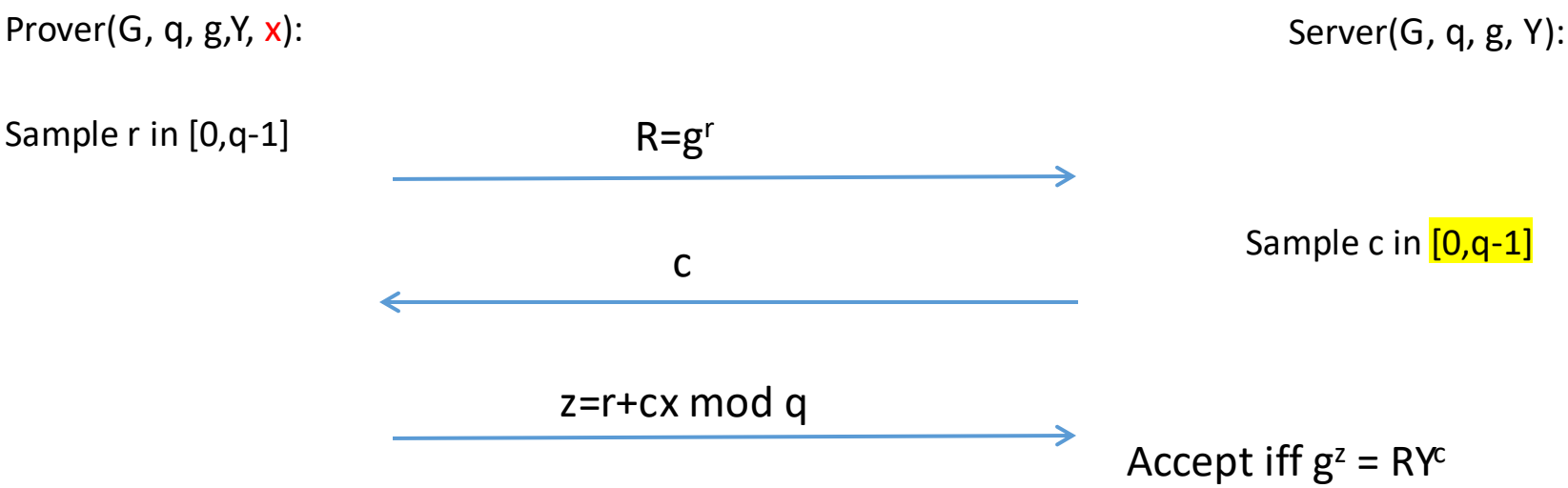
- A protocol involving a prover and a verifier.
- The prover takes as input a statement x and a witness w.
- The verifier takes as input the statement x.
- The prover's goal is to convince the verifier that some statement is true (or that she holds the witness w) without revealing any other information.
- Here, we only consider internal attackers, i.e., the prover will try to cheat the verifier and the verifier will try to learn the witness w.

- Correctness:
  - Completeness: Given honest prover and honest verifier, the protocol will output 1
- Security:
  - Soundness: If the statement is wrong (or the prover does not hold the witness), then she cannot pass the verification.
  - Zero-Knowledge: The verifier cannot learn any information from the protocol.
    - Here, we usually cannot use the indistinguishability-based definition (unless each statement is associated with multiple witnesses)
    - We use a simulation-based definition

Preliminaries on Cyclic Group

- Let (G,g,p) be a cyclic group, where G is the set of group element, g is the generator, and p is the group order:
  - $G=\{ g^0, g^1 , \ldots , g^{p-1}\}$
  - $g^p=1$
- The following operations are easy in the group (G,g,p):
  - Given any h_1, h_2 in G, it is easy to compute $h_1 \cdot h_2$
    - For any h in G and for any x,y in [0,p-1], given h^x and h^y, it is easy to compute $h^{x+y}=h^x \cdot h^y$
    - For any h_1, h_2 in G and for any x in[0,p-1], given h_1^x and h_2^x , we can compute $(h_1 \cdot h_2)^x= h_1^x \cdot h_2^x$
  - Given any h in G and any x in [0,p-1], it is easy to compute h^x
- The following operations are hard in the group (G,g,p):
  - Given g^x, it is hard to compute x (The DL problem)
  - Given g^x and g^y, it is hard to compute g^{xy} (The CDH problem)
  - Given g^x and g^y, it is hard to distinguish g^{xy} from a random group element in G (The DDH problem)

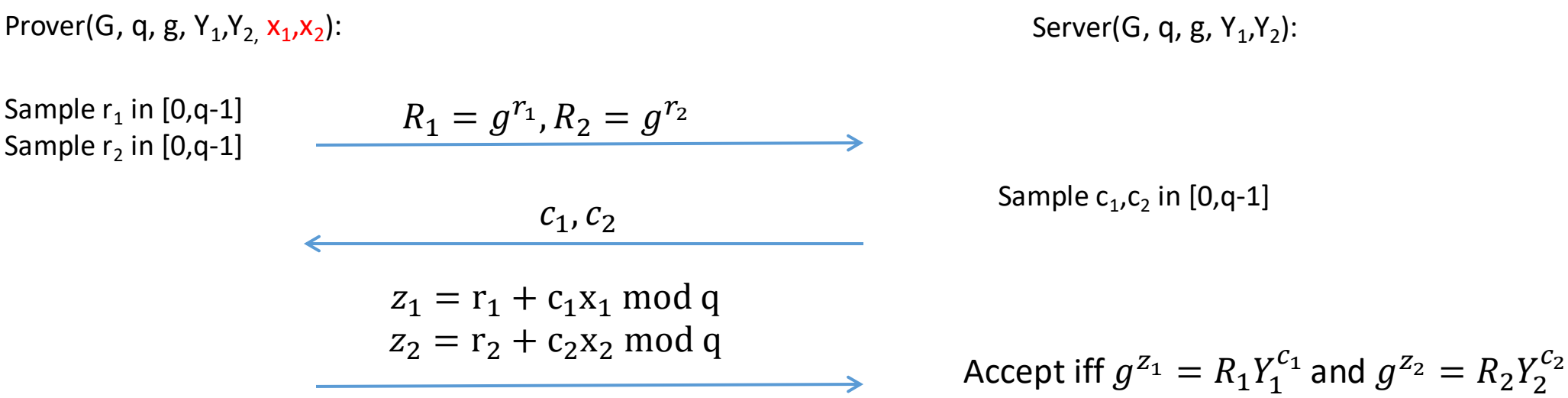
Schnorr Protocol



Zero-Knowledge Proofs for the AND relation in Cyclic Groups

A proof of knowledge of the And Relation

- How to prove that you know the discrete logs x_1, x_2 s.t. $g^{x_1} = Y_1$ and $g^{x_2} = Y_2$?
  - We can simply combine two protocols



A proof of knowledge of the And Relation

- How to prove that you know the discrete log x s.t. $g_1^x = Y_1$ and $g_2^x = Y_2$?
  - Can we combine two protocols directly?

A proof of knowledge of the And Relation

- How to prove that you know the discrete log x s.t. $g_1^x = Y_1$ and $g_2^x = Y_2$?
  - We need to add restrictions to ensure that the two discrete logs are the same.

Prover(G, q, g_1,g_2, Y_1,Y_2, x):

Server(G, q, g_1,g_2, Y_1,Y_2):

Sample r in [0,q-1]

$R_1 = g_1^r, R_2 = g_2^r$ →

Sample c in [0,q-1]

← c

$z = r + cx \bmod q$ →

Accept iff $g_1^z = R_1 Y_1^c$ and $g_2^z = R_2 Y_2^c$

A proof of knowledge of the And Relation

Prover(G, q, g_1,g_2, Y_1,Y_2, x):

Server(G, q, g_1,g_2, Y_1,Y_2):

Sample r in [0,q-1]

$$R_1 = g_1^r, R_2 = g_2^r$$

$$c$$

Sample c in [0,q-1]

$$z = r + cx \bmod q$$

Accept iff $g_1^z = R_1 Y_1^c$ and $g_2^z = R_2 Y_2^c$

- Completeness

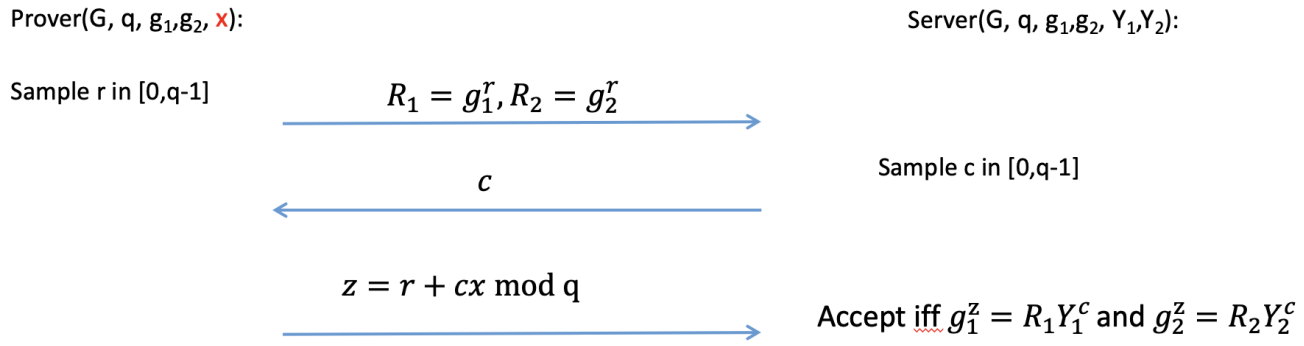
A proof of knowledge of the And Relation

Soundness

- Assume that the prover can always pass the verifications. Then after sending (R_1,R_2), the verifier is able to compute the correct response z on many challenges c.
- Now, based on two different challenges c_1 and c_2, and the correct responses z_1 and z_2, it is easy to extract *x*.
  - Given (R1,R2,c_1,z_1) and (R1,R2,c_2,z_2) where

$$g_1^{z_1} = R1 * Y_1^{c_1} \qquad g_1^{z_2} = R1 * Y_1^{c_2}$$
$$g_2^{z_1} = R2 * Y_2^{c_1} \qquad g_2^{z_2} = R2 * Y_2^{c_2}$$

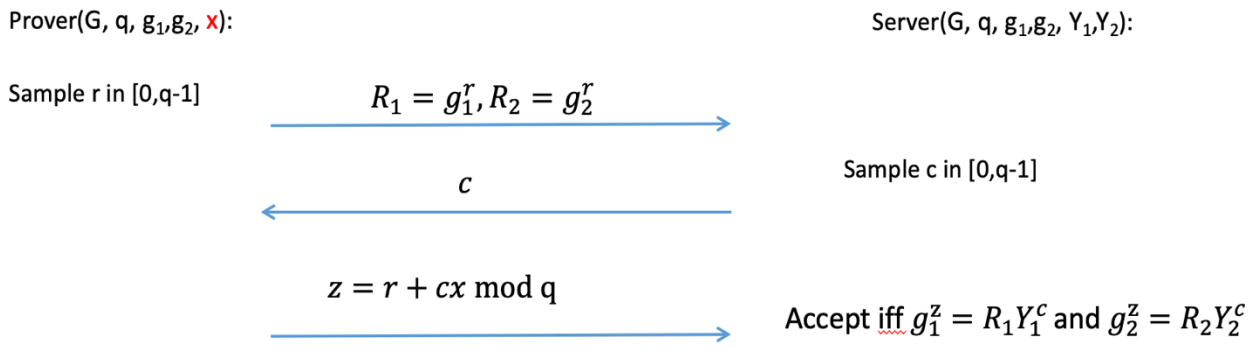
  - We have $g_1^{z_1-z_2} = Y_1^{c_1-c_2}$ and $g_2^{z_1-z_2} = Y_2^{c_1-c_2}$
  - Thus, one can compute x= $(z_1-z_2)(c_1-c_2)^{-1}$ which satisfies $Y_1 = g_1^{x} \wedge Y_2 = g_2^{x}$
- If the prover only passes with a non-negligible probability, a more detailed probabilty analysis and the rewinding techniques are needed.



A proof of knowledge of the And Relation

(Honest-Verifier) Zero-Knowledge

- The verifier only sees a random value
- Actually, one can **simulate the interaction** (R_1,R_2,c,Z) without knowing *x* as long as c is randomly chosen by the prover (i.e., the prover is honest).
  1. Choose a random c from Z_p
  2. Choose a random z
  3. Compute $R_1 = g_1^z * h_1^{-c}, R_2 = g_2^z * h_2^{-c}$(We set r=z-c*x mod q implicitly)



Zero-Knowledge Proofs for the OR relation in Cyclic Groups

A proof of knowledge of the OR Relation

- How to prove that you know the discrete log x s.t. either $g_1^x = Y_1$ or $g_2^x = Y_2$?
  - Assume that you know $g_1^x = Y_1$, then you can at least prove this statement.
  - It implies that you know x s.t. either $g_1^x = Y_1$ or $g_2^x = Y_2$, but it reveals for which part you know the witness.
  - To solve the problem, we need to also include a valid proof for the second part.
    - Fortunately, this is possible if we know the challenge in advance.
  - So, we need to design the protocol in a way that you can know one and only one challenge in advance.

A proof of knowledge of the OR Relation

- How to prove that you know the discrete log x s.t. either $g_1^x = Y_1$ or $g_2^x = Y_2$?

Prover(G, q, g_1,g_2, Y_1,Y_2, x):
W.L.O.G., assume that you know x satisfying $g_1^x = Y_1$

Server(G, q, g_1,g_2, Y_1,Y_2):

Sample r_1, c_2, z_2 in [0,q-1]
$R_1 = g_1^{r_1}, R_2 = g_2^{z_2}/Y_2^{c_2}$

R_1, R_2 →

Sample c in [0,q-1]

← c

$c_1 = c - c_2 \ mod \ q$
$z_1 = r_1 + c_1 x \ mod \ q$

c_1, c_2, z_1, z_2 →

Accept iff
$c = c_1 + c_2 \ mod \ q,$
$g_1^{z_1} = R_1 Y_1^{c_1}$,
$g_2^{z_2} = R_2 Y_2^{c_2}$

A proof of knowledge of the OR Relation

Prover(G, q, g_1,g_2, Y_1,Y_2, x):

W.L.O.G., assume that you know x satisfying $g_1^x = Y_1$

Server(G, q, g_1,g_2, Y_1,Y_2):

Sample r_1, c_2, z_2 in [0,q-1]

$R_1 = g_1^{r_1}, R_2 = g_2^{z_2}/Y_2^{c_2}$

R_1, R_2 →

← c

Sample c in [0,q-1]

$c_1 = c - c_2 \ mod \ q$

$z_1 = r_1 + c_1 x \ mod \ q$

c_1, c_2, z_1, z_2 →

Accept iff

$c = c_1 + c_2 \ mod \ q,$

$g_1^{z_1} = R_1 Y_1^{c_1}$,

$g_2^{z_2} = R_2 Y_2^{c_2}$

- Completeness

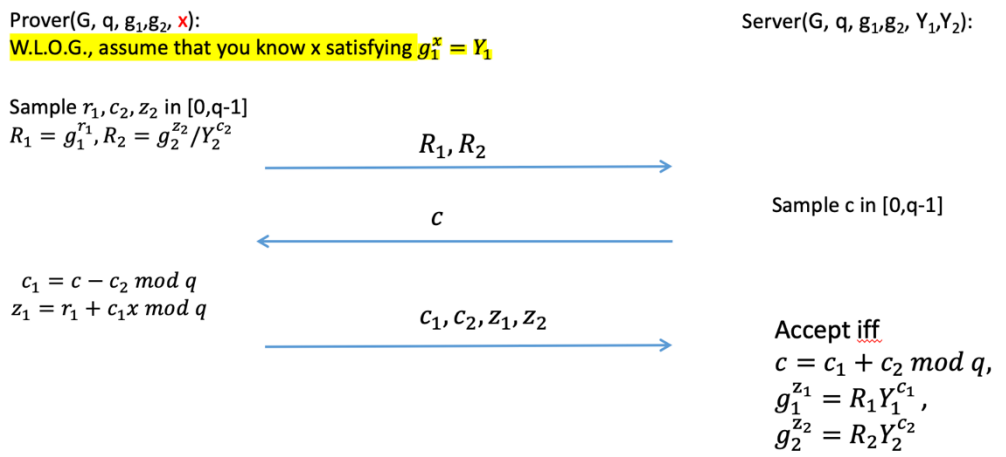
A proof of knowledge of the OR Relation

Soundness

- Assume that the prover can always pass the verifications. Then after sending (R_1,R_2), the verifier is able to compute the correct response on many challenges c.
- Now, based on two different challenges c and c', and the correct responses (c_1,c_2,z_1,z_2) and (c'_1,c'_2,z'_1,z'_2) , it is easy to extract *x*.
  - Given (R1,R2,c,c_1,c_2,z_1,z_2) and (R1,R2,c',c'_1,c'_2,z'_1,z'_2) where

$$g_1^{z_1} = R1 * h_1^{c_1} \qquad g_1^{z'_1} = R1 * h_1^{c'_1}$$
$$g_2^{z_2} = R2 * h_2^{c_2} \qquad g_2^{z'_2} = R2 * h_2^{c'_2}$$

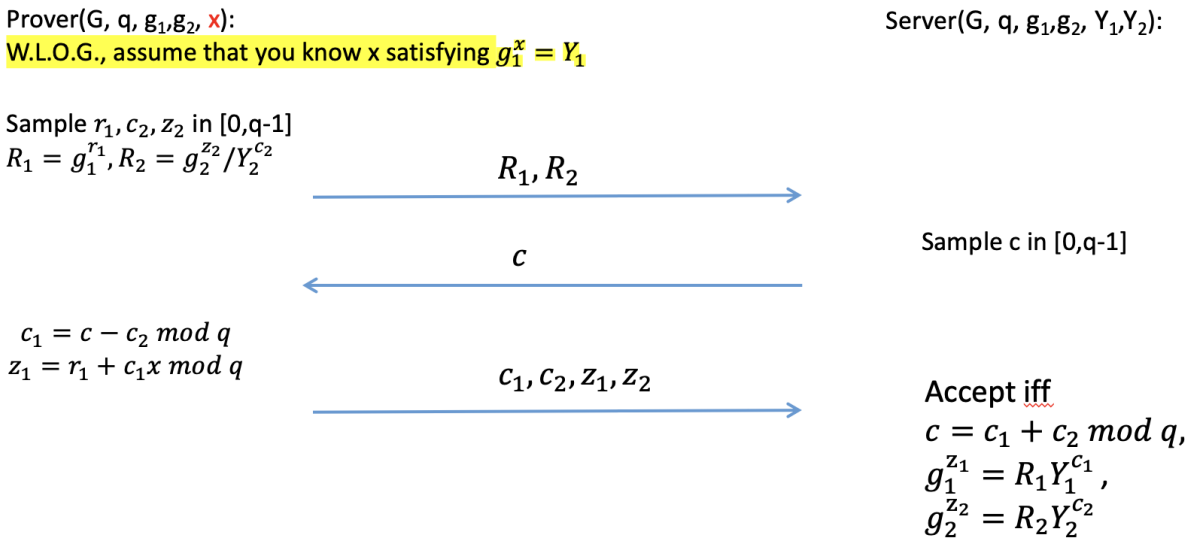
  - We have $g_1^{z_1 - z'_1} = h_1^{c_1 - c'_1}$ and $g_2^{z_2 - z'_2} = h_2^{c_2 - c'_2}$
  - As $c \neq c'$, we have either $c_1 \neq c'_1$ or $c_2 \neq c'_2$
  - Thus, one can compute either $x = (z_1 - z'_1)(c_1 - c'_1)^{-1}$ (which satisfies $Y_1 = g_1^x$) or $a = (z_2 - z'_2)(c_2 - c'_2)^{-1}$ (which satisfies $Y_2 = g_2^x$).
- If the prover only passes with a non-negligible probability, a more detailed probabilty analysis and the rewinding techniques are needed.



A proof of knowledge of the OR Relation

(Honest-Verifier) Zero-Knowledge

- The verifier only sees random values
- Actually, one can **simulate the interaction** (R_1,R_2,c, c_1,c_2,z_1,z_2) without knowing any *x* as long as c is randomly chosen by the prover (i.e., the prover is honest).
  1. Choose random c_1,c_2, from Z_p
  2. Compute c= c_1+c_2
  3. Choose random z_1,z_2
  4. Compute $R_1 = g_1^{z_1} * h_1^{-c_1}, R_2 = g_2^{z_2} * h_2^{-c_2}$



A proof of knowledge of the OR Relation

- Given $(g_1, Y_1), (g_2, Y_2), \ldots, (g_l, Y_l)$,how to prove that you know one of the discrete log, i.e., a number x s.t. $g_i^x = Y_i$?

Prover(G, q, $(g_1, Y_1), (g_2, Y_2), \ldots, (g_l, Y_l)$, x):
W.L.O.G., assume that you know x satisfying $g_1^x = Y_1$

Server(G, q, $(g_1, Y_1), (g_2, Y_2), \ldots, (g_l, Y_l)$):

Sample r_1 in [0,q-1], $R_1 = g_1^{r_1}$;
For i in [2,l]:
  Sample c_i, z_i in [0,q-1],
  $R_i = g_i^{z_i} / Y_i^{c_i}$

$R_1, R_2, \ldots R_l$ →

← c

Sample c in [0,q-1]

$c_1 = c - \sum_{i=2}^{l} c_i \bmod q$
$z_1 = r_1 + c_1 x \bmod q$

$(c_1, z_1), \ldots, (c_l, z_l)$ →

Accept iff
$c = \sum_{i=1}^{l} c_i \bmod q$,
$g_i^{z_i} = R_i Y_i^{c_i}$ for all i.

Ring Signature

Ring signatures

- In some cases, it is necessary to sign on a message while hiding the identity of the signer.
- Usually, we will require that the public only know that the signature is made by someone in a group, but they do not know the exact identity of the signer.
- For example, in some applications like anonymous reporting, we need to ensure that:
  - The whistleblower can sign the signature on behalf of a set of users (e.g., all staff in a company).
  - Anyone outside this set is not able to sign.
  - No one knows who is the signer/ whistleblower, i.e., the signatures produced by anyone in the set cannot be distinguished.
- The above scenario is a bit artificial, but we finally found some more natural application scenarios. (Assignment 2 Task 1)

Ring Signatures

- KeyGen(λ): Taking as input a security parameter λ, the key generation algorithms returns (pk,sk)

- Sign(sk, M, (pk_1, …, pk_l)): Taking as input a message M, a set of public keys, and a secret key sk for one of the public key, the signing algorithm returns a signature denoted by S.

$$S \leftarrow \text{Sign}(sk, M, (pk_1, \ldots, pk_l))$$

- Verify(S,M, (pk_1, …, pk_l)): Taking as input signed message (S,M) and the set of public keys, the verification algorithm returns 1 or 0.

Ring Signatures

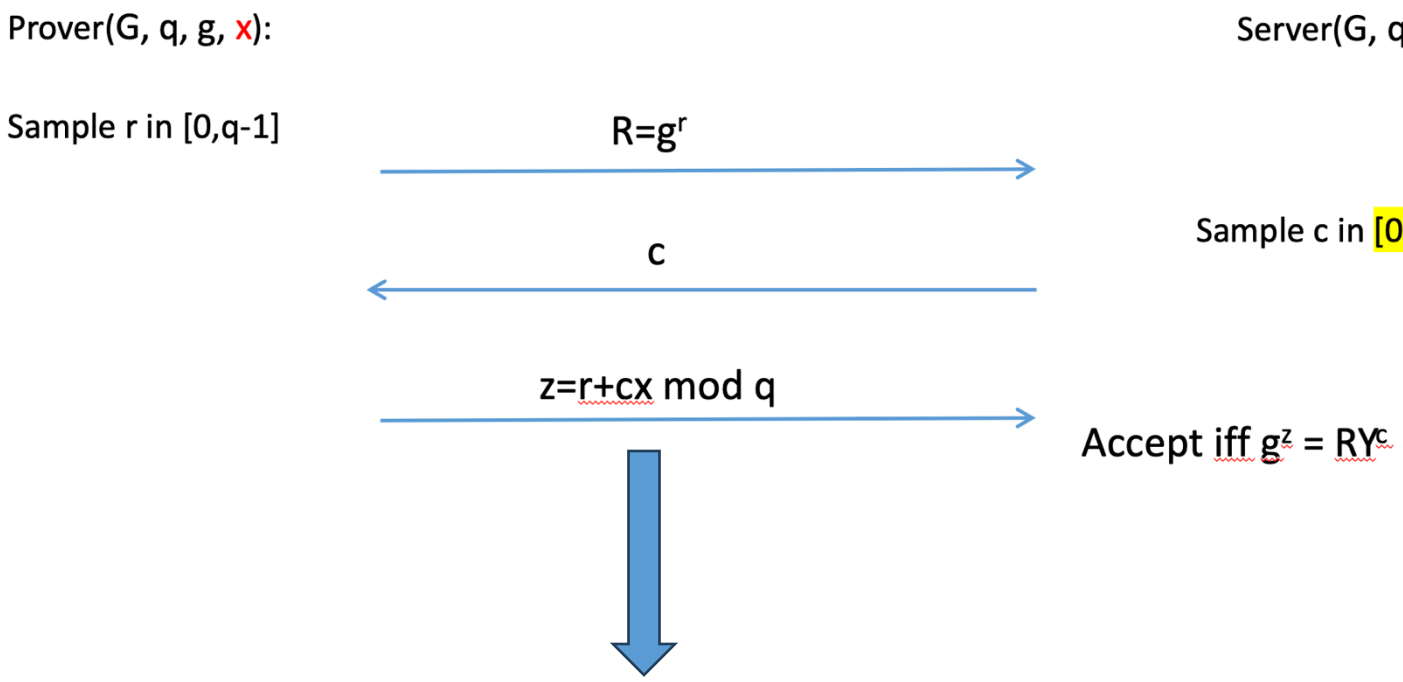
- **<u>Correctness</u>**: For all generated (pk_1,sk_1), …, (pk_l,sk_l), all index i, and all signature $S \leftarrow Sign(sk_i, M, (pk_1, \ldots, pk_l))$, we have

$$Pr[Verify(S,M, (pk_1, \ldots, pk_l))=1]=1$$

Ring Signatures

- Next, let us try to define the security. We need to define unforgeability (since it is a signature) and anonumity of Signer (since it is a ring signature).
- Unforgeability: Anyone outside the set cannot produce a valid signature.
  - The adversary should be able to
    - Aks for a signature on a message M singed by a secret key sk_i on behalf of a set of users.
    - Ask for the secret keys for public keys outside the target set.
  - The goal is to generate a valid signature on a message M* singed by a set R*, where
    - The adversary has not asked for the secret key for any public key in R*.
    - The adversary has not asked the signature for (M*,R*)
- Anonymity: The adversary cannot know the real signer in a group.
  - The adversary is able to know all public keys and secret keys.
    - In some definitions, we require the adversary cannot learn the secret keys of the two targets.
  - The adversary asks for a signature on a message M* and a ring R*, where the signature is singed by either sk_0 or sk_1; the adversary's goal is to distinguish which secret key is used.

Constructing a Ring signature: Warm-Up



In this transform, we transform a proof showing that "I know a secret key of the DL-based cryptosystem" into a signature.

- Sign(sk, M): Taking as input a message M and a secret key sk=(G, q, g, x, H), the P.P.T. algorithm
  1. Choose a random number r and computes $R=g^r$
  2. Compute c=H(R, M)
  3. Compute z=r+ c*x mod q
  4. The signature is (R,z)
- Verify(S,M,pk): Taking as input a signed message M, the public key pk=(G, q, g, h, H), and a signature (R,z), the P.P.T. algorithm
  1. Compute c'=H(R,M) and Accept the signature if $g^z=R\cdot h^{c'}$

Constructing a Ring signature: Warm-Up

Prover(G, q, $(g_1, Y_1), (g_2, Y_2), \dots, (g_l, Y_l)$, x):
W.L.O.G., assume that you know x satisfying $g_1^x = Y_1$

Server(G, q, $(g_1, Y_1), (g_2, Y_2), \dots, (g_l, Y_l)$):

Sample r_1 in [0,q-1], $R_1 = g_1^{r_1}$;
For i in [2,l]:
Sample c_i, z_i in [0,q-1],
$R_i = g_i^{z_i}/Y_i^{c_i}$

$R_1, R_2, \dots R_l$ →

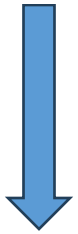
← c

Sample c in [0,q-1]

$c_1 = c - \sum_{i=2}^{l} c_i \bmod q$
$z_1 = r_1 + c_1 x \bmod q$

$(c_1, z_1), \dots, (c_l, z_l)$ →

Accept iff
$c = \sum_{i=1}^{l} c_i \bmod q$,
$g_i^{z_i} = R_i Y_i^{c_i}$ for all i.



The ZKP proves that "I know one secret key (out of l secret keys) of the DL-based cryptosystem". Can we transform it into a ring signature?

Ring Signature

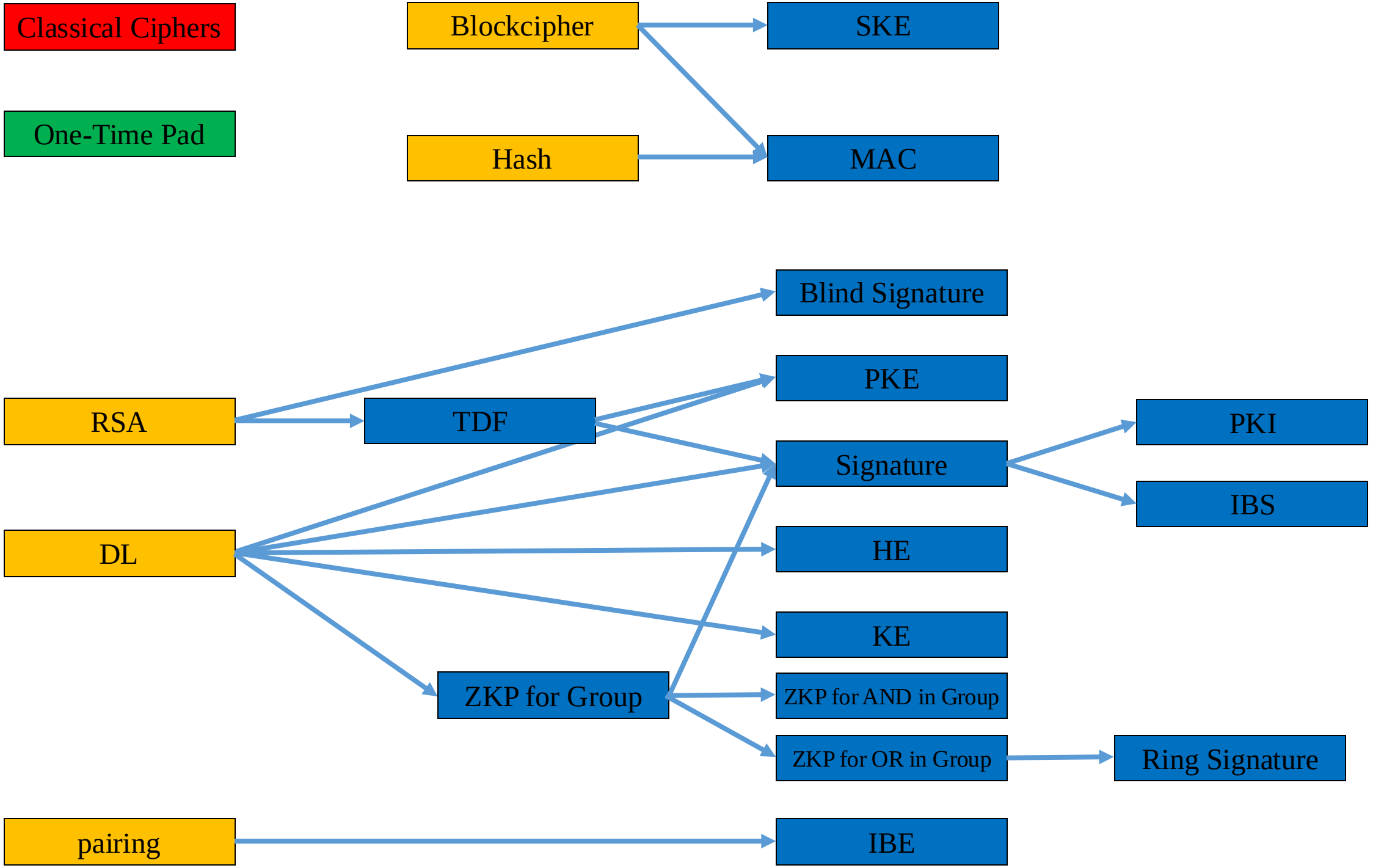
- The signature scheme assume that all parties agree on a cyclic group G of order q, a generator g of G, and a hash function H: {0,1}* $\rightarrow Z_q$

- KeyGen(λ): Taking as input a security parameter λ, the P.P.T. algorithm
  1. Chooses a uniform x ∈ Z_q and compute h = g^x .
  2. The public key is h and the private key is x.

- Sign(sk, M, (h_1, … h_l)): Taking as input a message M, a set of public keys (h_1, … h_l), and a secret key sk=x_i, the P.P.T. algorithm
  1. Choose a random number r_i
  2. Compute $R_i = g^{r_i}$
  3. For j in [1,l] and $j \neq i$:
     1. Choose random c_j, z_j
     2. Compute $R_j = g^{z_j}/h_j^{c_j}$
  4. Compute c=H(R_1, …, R_l, M)
  5. Compute $c_i = c - \sum_{j \neq i} c_j$
  6. Compute $z_i = r_i + c_i * x_i \; mod \; q$
  7. The signature is ((R_1,c_1,z_1), …, (R_l,c_l,z_l))

- Verify(S,M, (h_1, … h_l)): Taking as input a signed message M, a set of public keys (h_1, … h_l), and a signature ((R_1,c_1,z_1), …, (R_l,c_l,z_l)), the P.P.T. algorithm
  1. Compute c'=H(R_1, …, R_l, M)
  2. and Accept the signature if
     1. $c = \sum_{j=1}^{l} c_j \; mod \; q,$
     2. $g^{z_j} = R_j h_j^{c_j}$ for all j

- Correctness.
- Unforgeability
- Anonymity.


Classical Ciphers
One-Time Pad
Blockcipher
Hash
SKE
MAC
Blind Signature
PKE
RSA
TDF
PKI
Signature
IBS
DL
HE
KE
ZKP for Group
ZKP for AND in Group
ZKP for OR in Group
Ring Signature
pairing
IBE

Summary

- Zero-Knowledge
  - Zero-Knowledge for And Relation
    - Construction
    - Security*
  - Zero-Knowledge for OR Relation
    - General Idea
    - Construction
    - Security*
    - Extending to k statements
  - Ring Signature
    - Definition
      - Application Scenarios
      - Syntax and Correctness
      - Security*
    - Construction from ZKP for OR relations