Pairing-Based Cryptography
&
Generic Groups

Lecture 22

Bilinear Pairing

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups
  - $e(g^a, g^b) = e(g,g)^{ab}$

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups
  - $e(g^a,g^b) = e(g,g)^{ab}$
    - Multiplication (once) in the exponent!

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups
  - $e(g^a,g^b) = e(g,g)^{ab}$
    - Multiplication (once) in the exponent!
      - $e(g^a g^{a'},g^b) = e(g^a,g^b)\, e(g^{a'},g^b)$; $e(g^a,g^{bc}) = e(g^{ac},g^b)$; ...

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups
  - $e(g^a,g^b) = e(g,g)^{ab}$
    - Multiplication (once) in the exponent!
      - $e(g^a g^{a'},g^b) = e(g^a,g^b)\ e(g^{a'},g^b)$; $e(g^a,g^{bc}) = e(g^{ac},g^b)$; ...
    - Not degenerate: $e(g,g,) \neq 1$

Bilinear Pairing

- Two (or three) groups with an efficient pairing operation, $e: G \times G \rightarrow G_T$ that is "bilinear"
  - Typically, prime order (cyclic) groups
  - $e(g^a,g^b) = e(g,g)^{ab}$
    - Multiplication (once) in the exponent!
      - $e(g^a g^{a'},g^b) = e(g^a,g^b)\, e(g^{a'},g^b)$; $e(g^a,g^{bc}) = e(g^{ac},g^b)$; ...
    - Not degenerate: $e(g,g,) \neq 1$
- D-BDH Assumption: For random (a,b,c,z), the distributions of (g^a,g^b,g^c,g^{abc}) and (g^a,g^b,g^c,g^z) are indistinguishable

3-Party Key Exchange

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange
- Let $e: G \times G \rightarrow G_T$ be bilinear and g a generator of G

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange
- Let $e: G \times G \rightarrow G_T$ be bilinear and g a generator of G
- Alice broadcasts g^a, Bob broadcasts g^b, and Carol broadcasts g^c

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange
- Let $e: G \times G \rightarrow G_T$ be bilinear and g a generator of G
- Alice broadcasts g^a, Bob broadcasts g^b, and Carol broadcasts g^c
- Each party computes $e(g,g)^{abc}$

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange
- Let $e: G \times G \rightarrow G_T$ be bilinear and g a generator of G
- Alice broadcasts g^a, Bob broadcasts g^b, and Carol broadcasts g^c
- Each party computes $e(g,g)^{abc}$
  - e.g. Alice computes $e(g,g)^{abc} = e(g^b,g^c)^a$

3-Party Key Exchange

- A single round 3-party key-exchange protocol secure against passive eavesdroppers (under D-BDH assumption)
  - Generalizes Diffie-Hellman key-exchange
- Let $e: G \times G \rightarrow G_T$ be bilinear and g a generator of G
- Alice broadcasts g^a, Bob broadcasts g^b, and Carol broadcasts g^c
- Each party computes $e(g,g)^{abc}$
  - e.g. Alice computes $e(g,g)^{abc} = e(g^b,g^c)^a$
  - By D-BDH the key $e(g,g)^{abc} = e(g,g^{abc})$ is pseudorandom given eavesdropper's view (g^a,g^b,g^c)

NIZK Proofs

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)
  - Can forge proofs or extract knowledge if a trapdoor for the CRS is available (used by the simulator)

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)
  - Can forge proofs or extract knowledge if a trapdoor for the CRS is available (used by the simulator)
- NIZK useful in (non-interactive) public-key schemes

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)
  - Can forge proofs or extract knowledge if a trapdoor for the CRS is available (used by the simulator)
- NIZK useful in (non-interactive) public-key schemes
  - CRS can be part of the public key: when no security needed against the party generating CRS (e.g. signer of a message, receiver in an encryption scheme)

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)
  - Can forge proofs or extract knowledge if a trapdoor for the CRS is available (used by the simulator)
- NIZK useful in (non-interactive) public-key schemes
  - CRS can be part of the public key: when no security needed against the party generating CRS (e.g. signer of a message, receiver in an encryption scheme)
- Often "witness-indistinguishability" (NIWI or NIWI PoK) sufficient: can't distinguish proofs using different witnesses

NIZK Proofs

- Recall: ZK proofs to enforce honest behavior in a basic protocol (without compromising secrecy properties of the basic protocol)
- Non-interactive ZK, using a common random/reference string (CRS)
  - Can forge proofs or extract knowledge if a trapdoor for the CRS is available (used by the simulator)
- NIZK useful in (non-interactive) public-key schemes
  - CRS can be part of the public key: when no security needed against the party generating CRS (e.g. signer of a message, receiver in an encryption scheme)
- Often "witness-indistinguishability" (NIWI or NIWI PoK) sufficient: can't distinguish proofs using different witnesses
  - Trivial if only one witness. Very useful when two kinds of witnesses

NIZK Proofs

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation…") under fairly standard general assumptions

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation...") under fairly standard general assumptions
  - However, involves reduction to an NP-complete relation (e.g. graph Hamiltonicity) : considered impractical

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation…") under fairly standard general assumptions
  - However, involves reduction to an NP-complete relation (e.g. graph Hamiltonicity) : considered impractical
- Special purpose proof for statements that arise in specific schemes, under specific assumptions

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation…") under fairly standard general assumptions
  - However, involves reduction to an NP-complete relation (e.g. graph Hamiltonicity) : considered impractical
- Special purpose proof for statements that arise in specific schemes, under specific assumptions
  - Much more efficient: no NP-completeness reductions

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation...") under fairly standard general assumptions
  - However, involves reduction to an NP-complete relation (e.g. graph Hamiltonicity) : considered impractical
- Special purpose proof for statements that arise in specific schemes, under specific assumptions
  - Much more efficient: no NP-completeness reductions
    - e.g. Chaum-Pedersen Honest-Verifier ZK PoK of discrete log

NIZK Proofs

- NIZK proof/proof of knowledge systems exist for all "NP statements" (i.e., "there exists/I know a witness for the relation…") under fairly standard general assumptions
  - However, involves reduction to an NP-complete relation (e.g. graph Hamiltonicity) : considered impractical
- Special purpose proof for statements that arise in specific schemes, under specific assumptions
  - Much more efficient: no NP-completeness reductions
    - e.g. Chaum-Pedersen Honest-Verifier ZK PoK of discrete log
  - May exploit similar assumptions as used in the basic scheme

A NIZK For Statements Involving Pairings

A NIZK For Statements Involving Pairings

- Groth-Sahai proofs (2008)

A NIZK For Statements Involving Pairings

- Groth-Sahai proofs (2008)
- Very useful in constructions using bilinear pairings

A NIZK For Statements Involving Pairings

- Groth-Sahai proofs (2008)
- Very useful in constructions using bilinear pairings
- Can get "perfect" witness-indistinguishability or zero-knowledge

A NIZK For Statements Involving Pairings

- Groth-Sahai proofs (2008)
- Very useful in constructions using bilinear pairings
- Can get "perfect" witness-indistinguishability or zero-knowledge
  - Then, soundness will be under certain computational assumptions

A NIZK For Statements Involving Pairings

A NIZK For Statements Involving Pairings

- an e.g. statement

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know X,Y,Z $\in$ G and integers u,v,w s.t.

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know $X,Y,Z \in G$ and integers u,v,w s.t.
    - $e(X,A) \ldots e(X,Y) = 1$ (pairing product)

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know $X,Y,Z \in G$ and integers u,v,w s.t.
    - $e(X,A) \ldots e(X,Y) = 1$ (pairing product)
    - $X^{au} \ldots Z^{bv} = B$ (product)

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know $X,Y,Z \in G$ and integers u,v,w s.t.
    - $e(X,A) \ldots e(X,Y) = 1$ (pairing product)
    - $X^{au} \ldots Z^{bv} = B$ (product)
    - $a\,v + \ldots + b\,w = c$

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know $X,Y,Z \in G$ and integers u,v,w s.t.
    - $e(X,A) \ldots e(X,Y) = 1$ (pairing product)
    - $X^{au} \ldots Z^{bv} = B$ (product)
    - $a\,v + \ldots + b\,w = c$
  - (where $A,B \in G$, integers a,b,c are known to both)

A NIZK For Statements Involving Pairings

- an e.g. statement
  - I know $X,Y,Z \in G$ and integers u,v,w s.t.
    - $e(X,A) \ldots e(X,Y) = 1$ (pairing product)
    - $X^{au} \ldots Z^{bv} = B$ (product)
    - $a\,v + \ldots + b\,w = c$
  - (where $A,B \in G$, integers a,b,c are known to both)
- Useful in proving statements like "these two commitments are to the same value", or "I have a signature for a message with a certain property", when appropriate commitment/signature scheme is used

Applications

Applications

- Fancy signature schemes

Applications

- Fancy signature schemes
  - Short group/ring signatures

Applications

- Fancy signature schemes
  - Short group/ring signatures
  - Short attribute-based signatures

Applications

- Fancy signature schemes
  - Short group/ring signatures
  - Short attribute-based signatures
- Efficient non-interactive proof of correctness of shuffle

Applications

- Fancy signature schemes
  - Short group/ring signatures
  - Short attribute-based signatures
- Efficient non-interactive proof of correctness of shuffle
- Non-interactive anonymous credentials

Applications

- Fancy signature schemes
  - Short group/ring signatures
  - Short attribute-based signatures
- Efficient non-interactive proof of correctness of shuffle
- Non-interactive anonymous credentials
- ...

Some More Assumptions

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}

- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}
- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)
  - q-SDH: Given $(g,g^x,\ldots,g^{x\hat{}q})$, infeasible to find $(y,g^{1/x+y})$

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}
- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)
  - q-SDH: Given $(g,g^x,\ldots,g^{x\hat{}q})$, infeasible to find $(y,g^{1/x+y})$
- Decision-Linear Assumption: $(g,g^a,g^b,g^{ax},g^{by}, g^{x+y})$ and $(g,g^a,g^b,g^{ax},g^{by}, g^z)$ are indistinguishable

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}
- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)
  - q-SDH: Given $(g,g^x,\ldots,g^{x\hat{}q})$, infeasible to find $(y,g^{1/x+y})$
- Decision-Linear Assumption: $(g,g^a,g^b,g^{ax},g^{by}, g^{x+y})$ and $(g,g^a,g^b,g^{ax},g^{by}, g^z)$ are indistinguishable
- Variants and other assumptions, in different settings

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}
- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)
  - q-SDH: Given $(g,g^x,\ldots,g^{x^\wedge q})$, infeasible to find $(y,g^{1/x+y})$
- Decision-Linear Assumption: $(g,g^a,g^b,g^{ax},g^{by}, g^{x+y})$ and $(g,g^a,g^b,g^{ax},g^{by}, g^z)$ are indistinguishable
- Variants and other assumptions, in different settings
  - When $e:G_1 x G_2 \rightarrow G_T$: DDH in G_1 and/or G_2

Some More Assumptions

- C-BDH Assumption: For random (a,b,c), given (g^a,g^b,g^c) infeasible to compute g^{abc}
- Strong DH Assumption: For random x, given (g,g^x) infeasible to find $(y,g^{1/x+y})$. (But can check: $e(g^x g^y, g^{1/x+y}) = e(g,g)$.)
    - q-SDH: Given $(g,g^x,\ldots,g^{x\hat{}q})$, infeasible to find $(y,g^{1/x+y})$
- Decision-Linear Assumption: $(g,g^a,g^b,g^{ax},g^{by}, g^{x+y})$ and $(g,g^a,g^b,g^{ax},g^{by}, g^z)$ are indistinguishable
- Variants and other assumptions, in different settings
    - When $e:G_1 x G_2 \rightarrow G_T$: DDH in G_1 and/or G_2
    - When G has composite order: Pseudorandomness of random elements from a prime order subgroup of G.

Cheap Crypto

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked
- Sometimes the resulting schemes may be quite complicated and relatively inefficient

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked
- Sometimes the resulting schemes may be quite complicated and relatively inefficient
- Quicker/cheaper alternative: Use heuristic idealizations

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked
- Sometimes the resulting schemes may be quite complicated and relatively inefficient
- Quicker/cheaper alternative: Use heuristic idealizations
  - Random Oracle Model

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked
- Sometimes the resulting schemes may be quite complicated and relatively inefficient
- Quicker/cheaper alternative: Use heuristic idealizations
  - Random Oracle Model
  - Generic Group Model

Cheap Crypto

- A significant amount of effort/expertise required to reduce the security to (standard) hardness assumptions
  - Or even to new "simple" assumptions
  - New assumptions may not have been actively attacked
- Sometimes the resulting schemes may be quite complicated and relatively inefficient
- Quicker/cheaper alternative: Use heuristic idealizations
  - Random Oracle Model
  - Generic Group Model
- Useful in at least "prototyping" new primitives (e.g. IBE)

Generic Group Model

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:
    - Sample: pick random x and return Handle(x)

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:
    - Sample: pick random x and return Handle(x)
    - Multiply: On input two handles h_1 and h_2, return Handle(Elem(h_1).Elem(h_2))

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:
    - Sample: pick random x and return Handle(x)
    - Multiply: On input two handles h_1 and h_2, return Handle(Elem(h_1).Elem(h_2))
    - Raise: On input a handle h and integer a (can be negative), return Handle(Elem(h)^a)

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:
    - Sample: pick random x and return Handle(x)
    - Multiply: On input two handles h_1 and h_2, return Handle(Elem(h_1).Elem(h_2))
    - Raise: On input a handle h and integer a (can be negative), return Handle(Elem(h)^a)
  - In addition, if modeling a group with bilinear pairing, also provides the pairing operation and operations for the target group

Generic Group Model

- A group is modeled as an oracle, which uses "handles" to represent group elements
  - The oracle maintains an internal table mapping group elements to handles one-to-one. Handles are generated arbitrarily in response to queries (say, randomly, or "symbolically")
  - Provides the following operations:
    - Sample: pick random x and return Handle(x)
    - Multiply: On input two handles h_1 and h_2, return Handle(Elem(h_1).Elem(h_2))
    - Raise: On input a handle h and integer a (can be negative), return Handle(Elem(h)^a)
  - In addition, if modeling a group with bilinear pairing, also provides the pairing operation and operations for the target group
- Discrete-log assumption, DDH (or B-DDH), DLin etc. are true in GGM

Generic Group Model

Generic Group Model

- Cryptographic scheme will be defined in the generic group model

Generic Group Model

- Cryptographic scheme will be defined in the generic group model
- Typically an underlying group of exponentially large order

Generic Group Model

- Cryptographic scheme will be defined in the generic group model
- Typically an underlying group of exponentially large order
- Adversary knows the underlying group structure, and may perform unlimited computations, but is allowed to query the oracle only a polynomial number of times over all

Generic Group Model

- Cryptographic scheme will be defined in the generic group model
- Typically an underlying group of exponentially large order
- Adversary knows the underlying group structure, and may perform unlimited computations, but is allowed to query the oracle only a polynomial number of times over all
- Can write the discrete log of every handle as a linear polynomial (or a quadratic polynomial, if allowing pairing) in variables corresponding to the sampling operation. An "accidental collision" if two formally different polynomials give same value

Generic Group Model

- Cryptographic scheme will be defined in the generic group model
- Typically an underlying group of exponentially large order
- Adversary knows the underlying group structure, and may perform unlimited computations, but is allowed to query the oracle only a polynomial number of times over all
- Can write the discrete log of every handle as a linear polynomial (or a quadratic polynomial, if allowing pairing) in variables corresponding to the sampling operation. An "accidental collision" if two formally different polynomials give same value
    - Negligible probability of accidental collision: by "Schwartz-Zippel Lemma", number of zeroes of a (non-zero) low-degree multi-variate polynomial is bounded

Generic Group Model

- Cryptographic scheme will be defined in the generic group model
- Typically an underlying group of exponentially large order
- Adversary knows the underlying group structure, and may perform unlimited computations, but is allowed to query the oracle only a polynomial number of times over all
- Can write the discrete log of every handle as a linear polynomial (or a quadratic polynomial, if allowing pairing) in variables corresponding to the sampling operation. An "accidental collision" if two formally different polynomials give same value
  - Negligible probability of accidental collision: by "Schwartz-Zippel Lemma", number of zeroes of a (non-zero) low-degree multi-variate polynomial is bounded
  - And an exhaustive analysis in terms of formal polynomials to show requisite security properties

Generic Group Model

Generic Group Model

- What does security in GGM mean?

Generic Group Model

- What does security in GGM mean?
- Secure against adversaries who do not "look inside" the group

Generic Group Model

- What does security in GGM mean?
- Secure against adversaries who do not "look inside" the group
- Risk: There maybe a simple attack against our construction because of some specific (otherwise benign) structure in the group

Generic Group Model

- What does security in GGM mean?
- Secure against adversaries who do not "look inside" the group
- Risk: There maybe a simple attack against our construction because of some specific (otherwise benign) structure in the group
  - No "if this scheme is broken, so are many others" guarantee

Generic Group Model

- What does security in GGM mean?
- Secure against adversaries who do not "look inside" the group
- Risk: There maybe a simple attack against our construction because of some specific (otherwise benign) structure in the group
  - No "if this scheme is broken, so are many others" guarantee
- Better practice: when possible identify simple (new) assumptions sufficient for the security of the scheme. Then prove the assumption in the generic group model

“Knowledge” Assumptions

"Knowledge" Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" b

“Knowledge” Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it “must know” b
- KEA-3: Given (g,g^a,g^b,g^{ab}) for random g,a,b, if a PPT adversary outputs (h,h^b), then it “must know” c_1, c_2 such that $h=g^{c1} (g^a)^{c2}$ (and $h^b=(g^b)^{c1} (g^{ab})^{c2}$)

“Knowledge” Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it “must know” b
- KEA-3: Given (g,g^a,g^b,g^{ab}) for random g,a,b, if a PPT adversary outputs (h,h^b), then it “must know” c_1, c_2 such that $h=g^{c1} (g^a)^{c2}$ (and $h^b=(g^b)^{c1} (g^{ab})^{c2}$)
  - By “fixing” KEA-2 (which forgot to consider c_1)

"Knowledge" Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" b

- KEA-3: Given (g,g^a,g^b,g^{ab}) for random g,a,b, if a PPT adversary outputs (h,h^b), then it "must know" c_1, c_2 such that $h=g^{c1} (g^a)^{c2}$ (and $h^b=(g^b)^{c1} (g^{ab})^{c2}$)
  - By "fixing" KEA-2 (which forgot to consider c_1)

- KEA-DH: Given g, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" either a or b

"Knowledge" Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" b
- KEA-3: Given (g,g^a,g^b,g^{ab}) for random g,a,b, if a PPT adversary outputs (h,h^b), then it "must know" c_1, c_2 such that $h=g^{c1} (g^a)^{c2}$ (and $h^b=(g^b)^{c1} (g^{ab})^{c2}$)
  - By "fixing" KEA-2 (which forgot to consider c_1)
- KEA-DH: Given g, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" either a or b
- All provable in the generic group model (for g with large order)

"Knowledge" Assumptions

- KEA-1: Given (g,g^a) for a random generator g and random a, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" b
- KEA-3: Given (g,g^a,g^b,g^{ab}) for random g,a,b, if a PPT adversary outputs (h,h^b), then it "must know" c_1, c_2 such that $h=g^{c1} (g^a)^{c2}$ (and $h^b=(g^b)^{c1} (g^{ab})^{c2}$)
  - By "fixing" KEA-2 (which forgot to consider c_1)
- KEA-DH: Given g, if a PPT adversary extends it to a DDH tuple (g,g^a,g^b,g^{ab}) then it "must know" either a or b
- All provable in the generic group model (for g with large order)
  - Even if the group has a bilinear pairing operation

Today

Today

- Bilinear Pairings

Today

- Bilinear Pairings
  - D-BDH and Joux's 3-party key-exchange

Today

- Bilinear Pairings
  - D-BDH and Joux's 3-party key-exchange
  - Groth-Sahai NIZK/NIWI proofs/PoKs

Today

- Bilinear Pairings
  - D-BDH and Joux's 3-party key-exchange
  - Groth-Sahai NIZK/NIWI proofs/PoKs
  - Various recent assumptions used

Today

- Bilinear Pairings
  - D-BDH and Joux's 3-party key-exchange
  - Groth-Sahai NIZK/NIWI proofs/PoKs
  - Various recent assumptions used
- Generic Group Model

Today

- Bilinear Pairings
  - D-BDH and Joux's 3-party key-exchange
  - Groth-Sahai NIZK/NIWI proofs/PoKs
  - Various recent assumptions used
- Generic Group Model
  - Knowledge-of-Exponent Assumptions