Generalization and Modularization of the ACCE Model

SKECH Workshop

2018-06-11

Horst Görtz Institute for IT Security
Chair for Network and Data Security

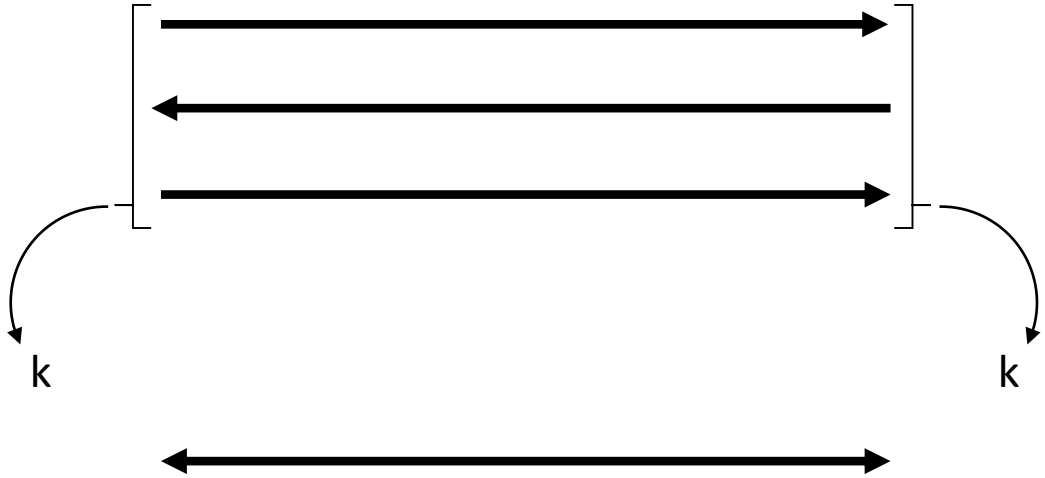
Benjamin Dowling, **Paul Rösler**, Jörg Schwenk

Agenda

- Key Exchange + Channel = ?
- Generalization of ACCE
- Modularization of ACCE
- Application to Noise

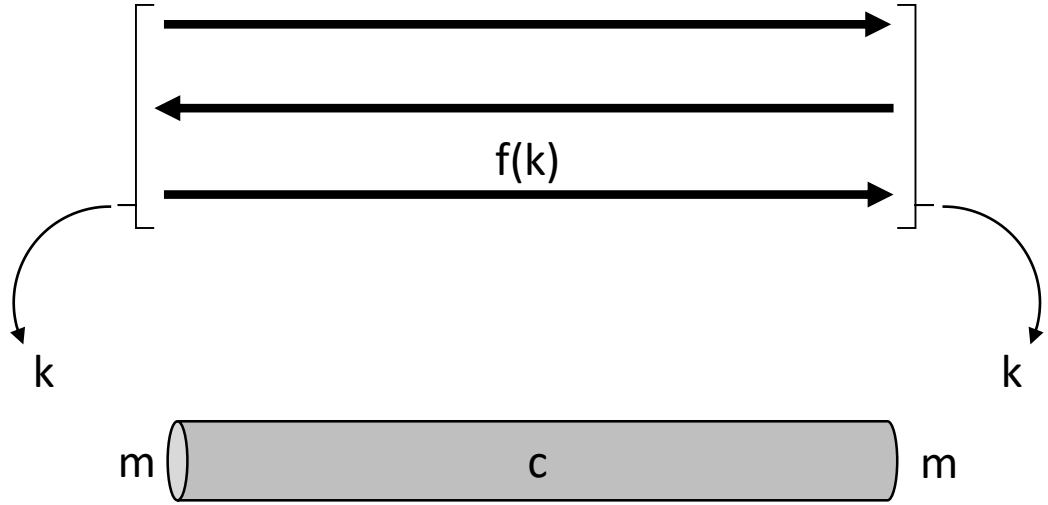
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*



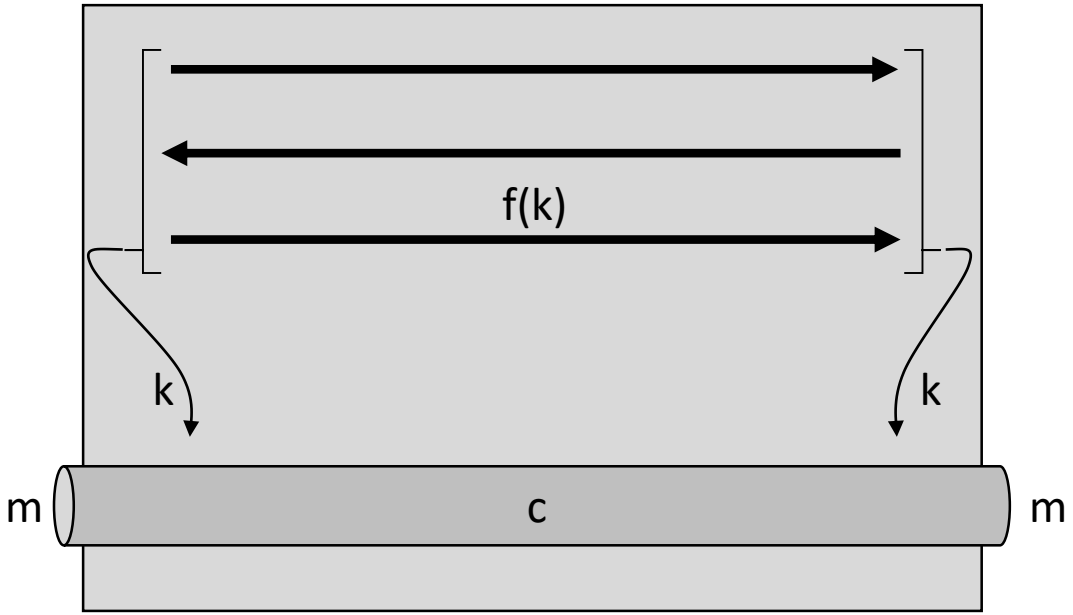
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*



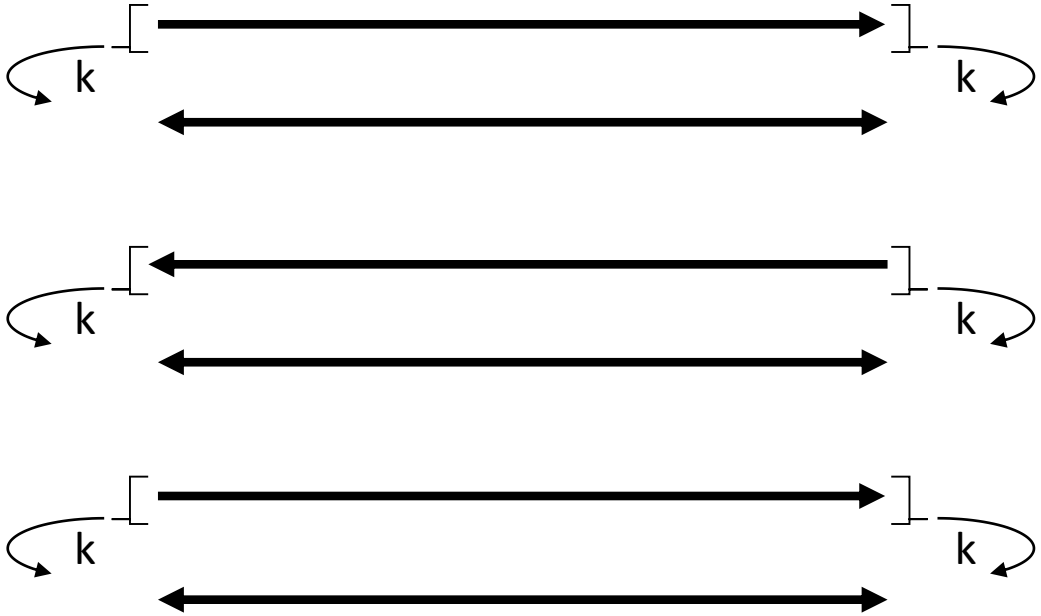
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*



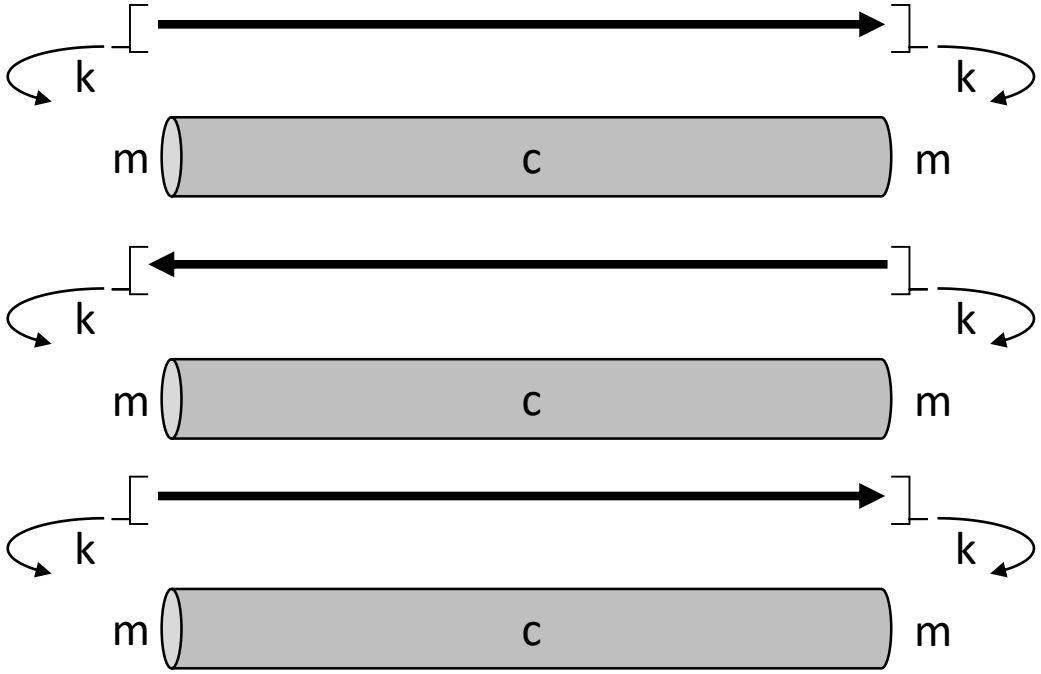
Key Exchange + Channel = ?

- Key exchange **then** symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange **and** symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol CCS14*



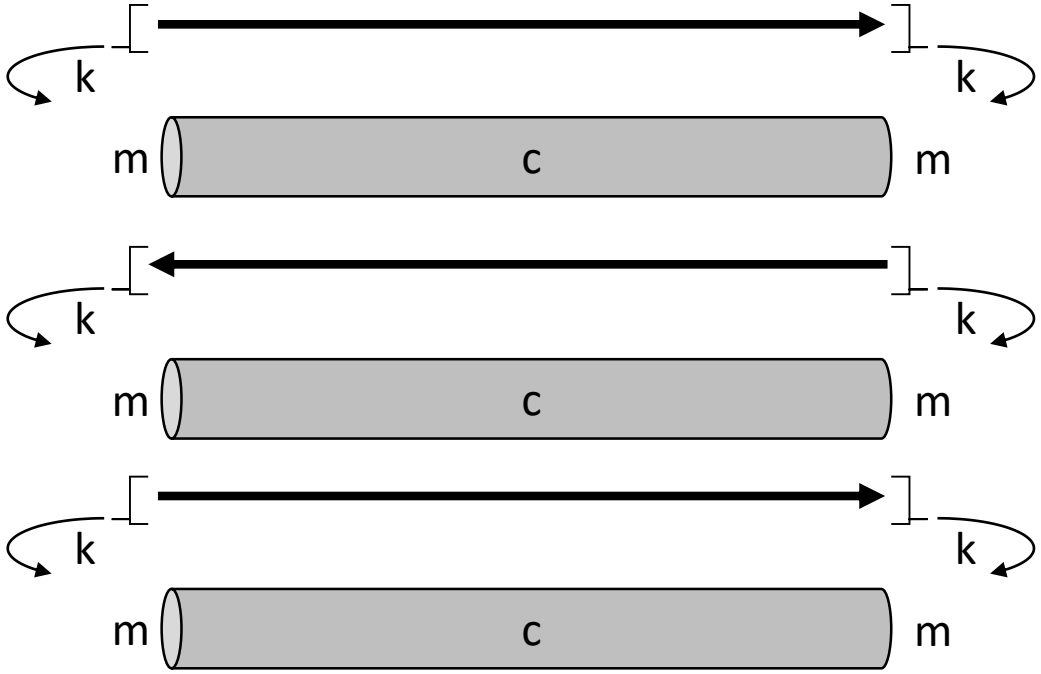
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol CCS14*



Key Exchange + Channel = ?

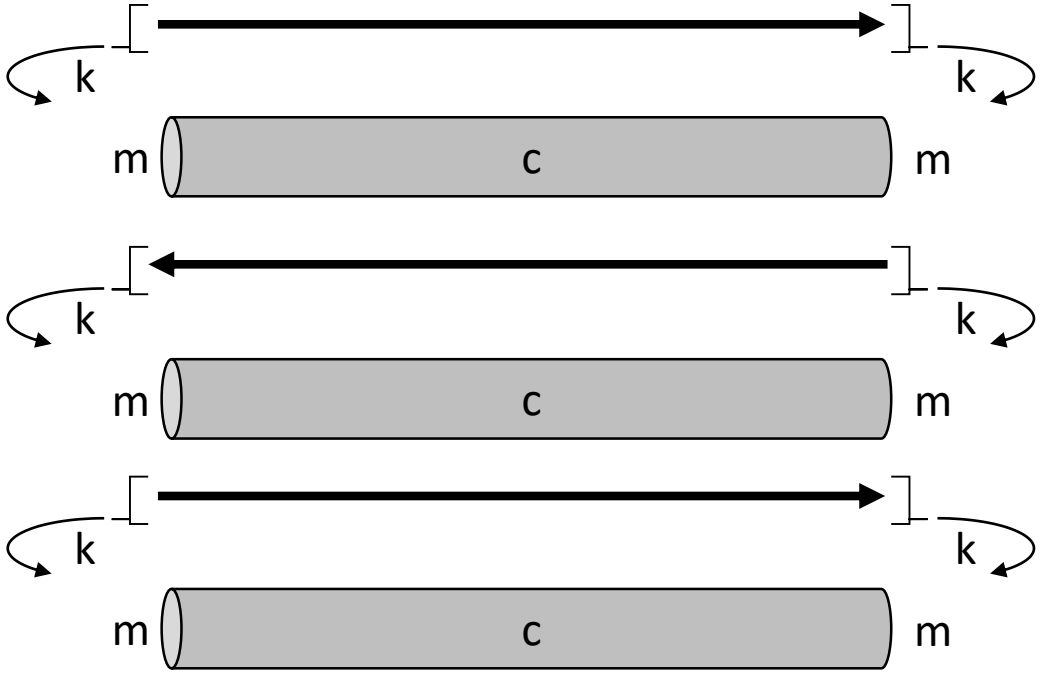
- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol CCS14*



Authentication from first message

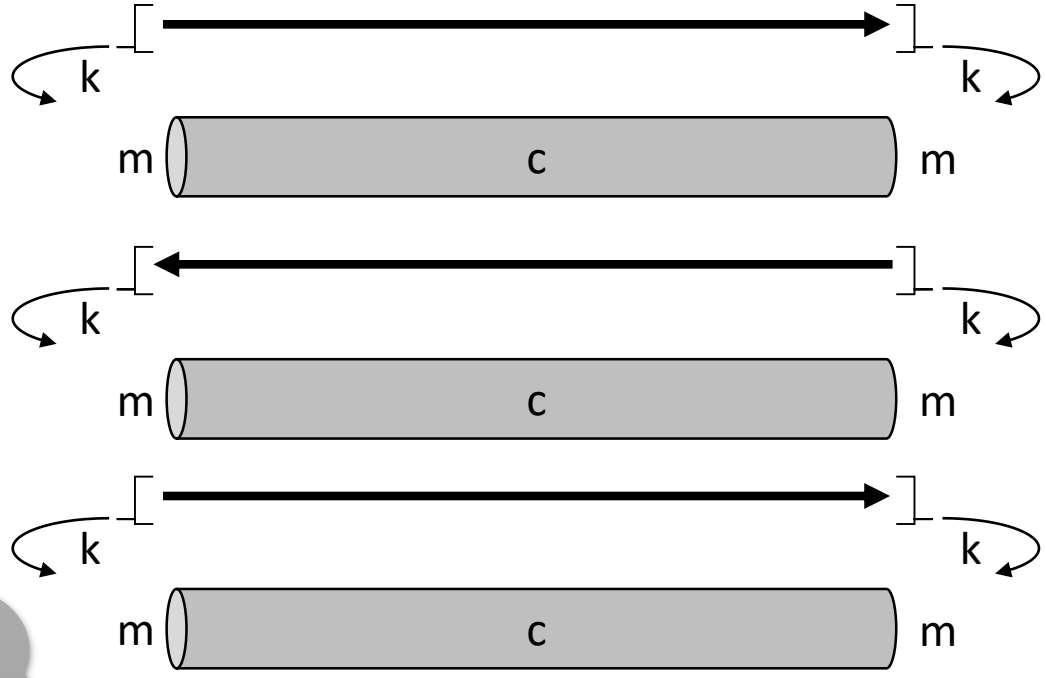
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol + **DFGS15***



Key Exchange + Channel = ?

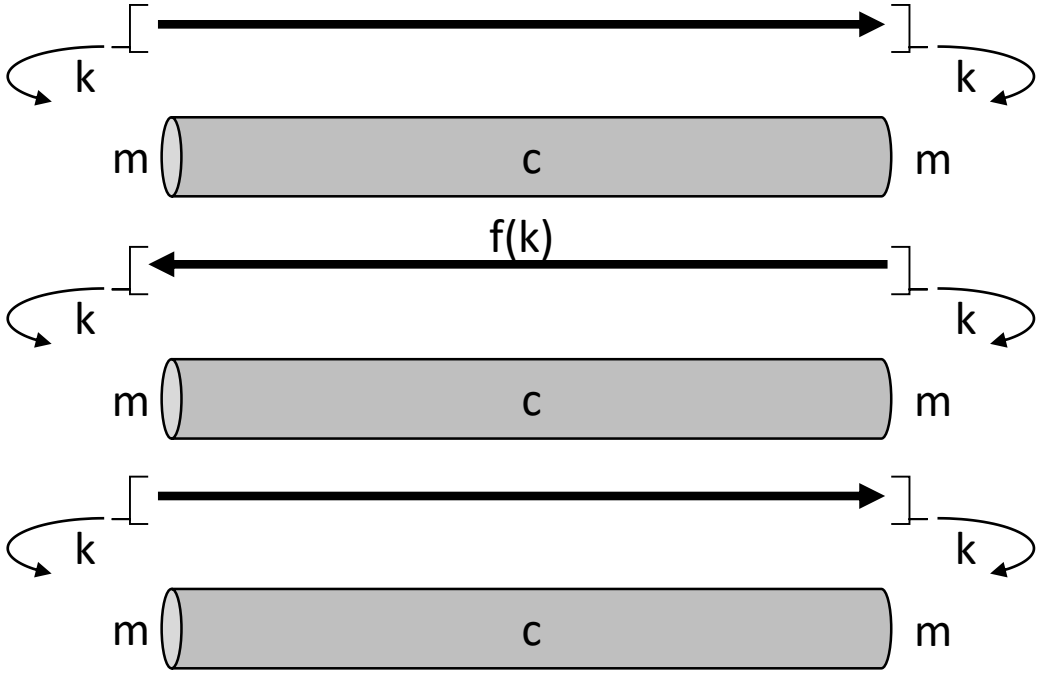
- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol + DFGS15 +* ***FG17***



Replay attacks allowed, internal keys…?!

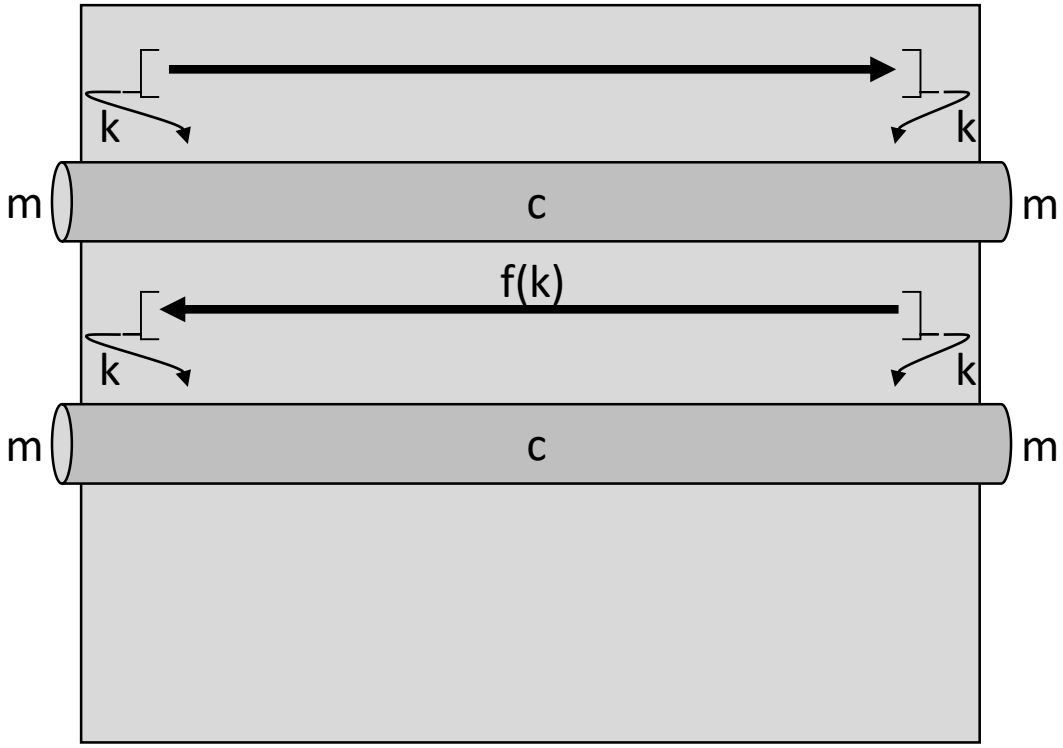
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol + DFGS15 + FG17*



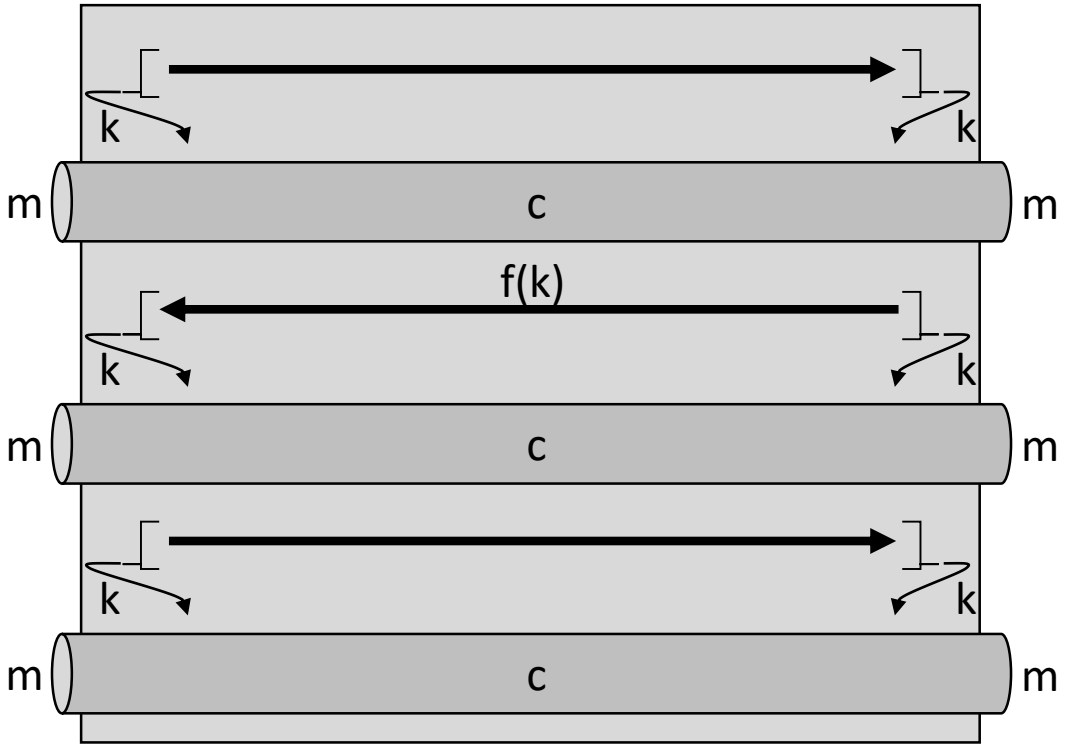
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol + DFGS15 + FG17*
- Two stage channel establishment
  - *Lychev et al.: How Secure and Quick is QUIC? Provable Security and Performance Analyses S&P15*



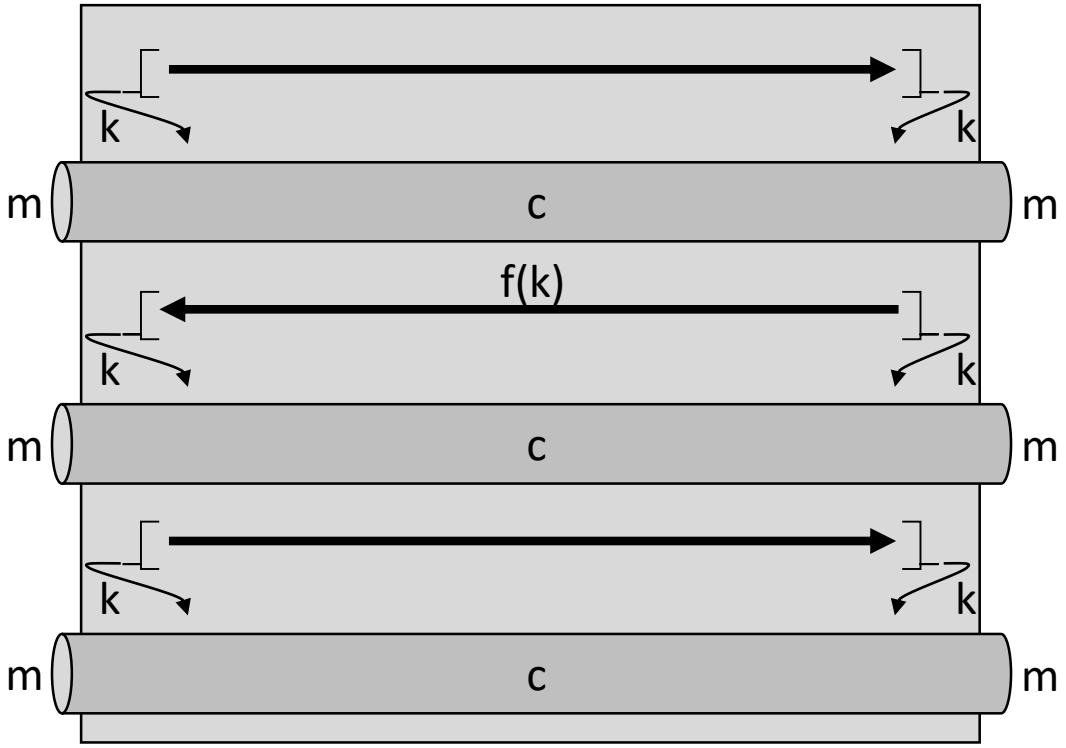
Key Exchange + Channel = ?

- Key exchange then symmetric protocol
  - *Brzuska et al.: Composability of Bellare-Rogaway Key Exchange Protocols CCS11*
- Channel establishment
  - *Jager et al.: On the Security of TLS-DHE in the Standard Model C12*
- Key exchange and symmetric protocol
  - *Fischlin, Günther: Multi-Stage Key Exchange and the Case of Google's QUIC Protocol + DFGS15 + FG17*
- Two stage channel establishment
  - *Lychev et al.: How Secure and Quick is QUIC? Provable Security and Performance Analyses S&P15*
- What is so new about it?



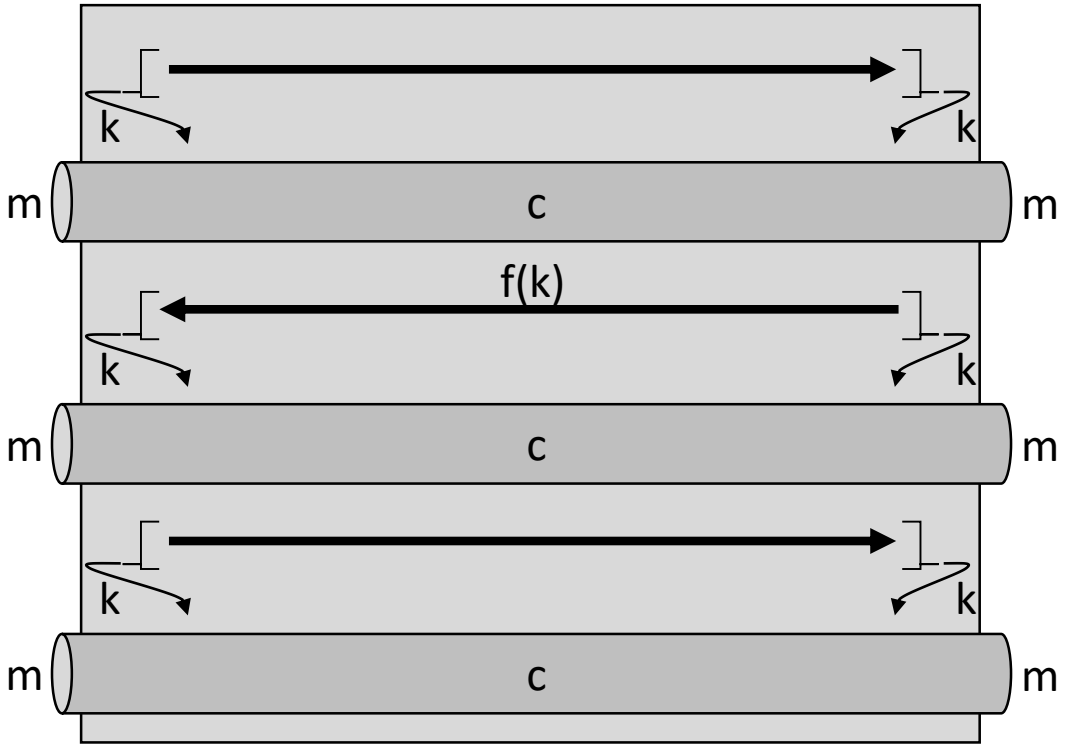
Generic and Modular ACCE

- What is so new about it?
  - Generic model (i.e., independent of analyzed protocol)



Generic and Modular ACCE

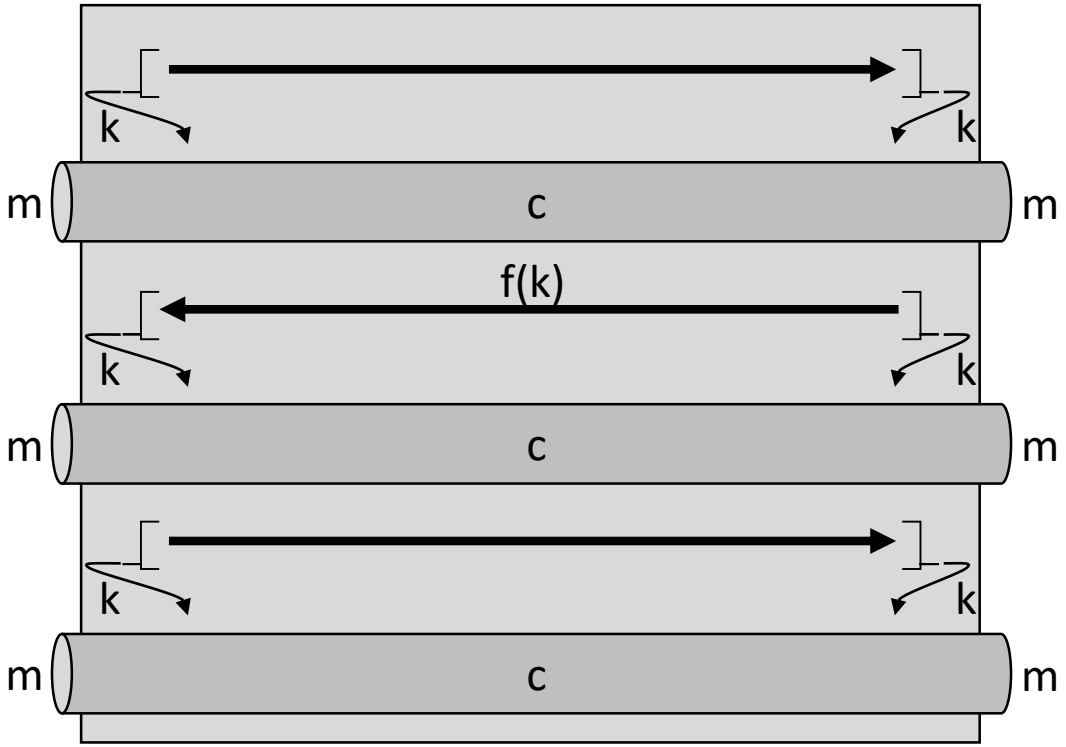
- What is so new about it?
  - Generic model (i.e., independent of analyzed protocol)
  - Channel security under key usage in *KE*, full modularity for security properties



Generic and Modular ACCE

- What is so new about it?
  - Generic model (i.e., independent of analyzed protocol)
  - Channel security under key usage in *KE*, full modularity for security properties
  - Allows to analyze protocols as they are
    - Signal*
    - Noise

    → Wireguard



* Composition of X3DH and DRAlg?

Key Exchange + Channel = ?

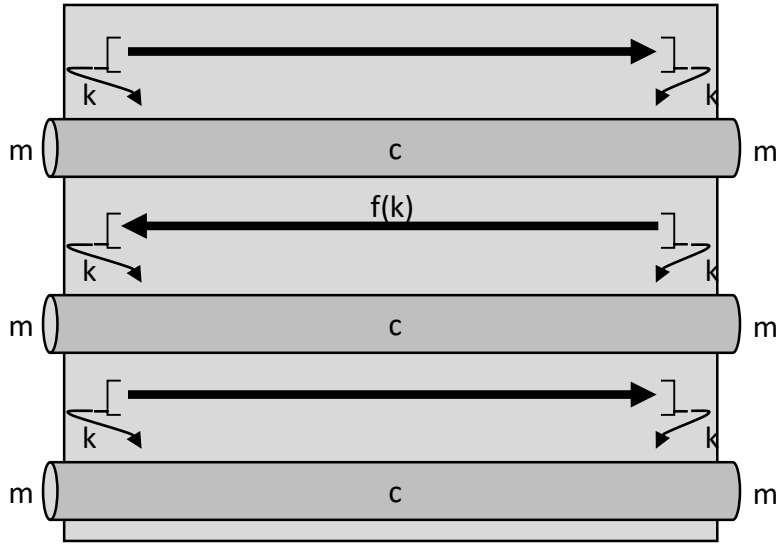
Generalization of ACCE

Modularization of ACCE

Application to Noise

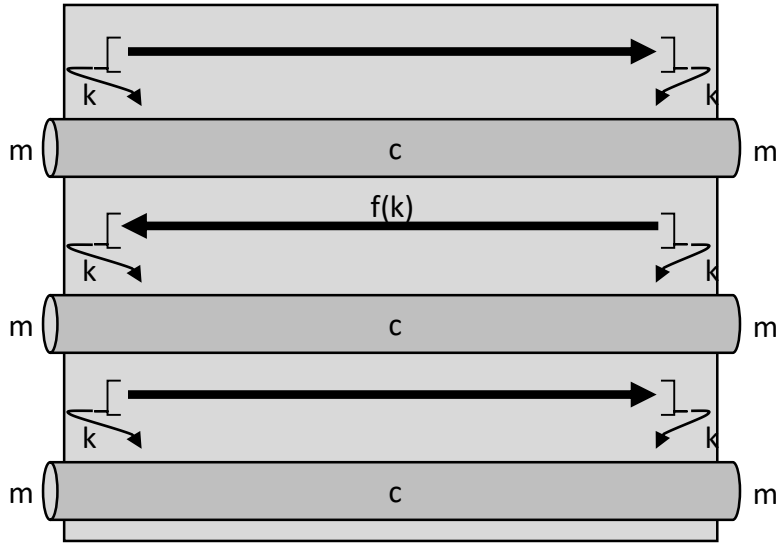
Generalization of ACCE

- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



Generalization of ACCE

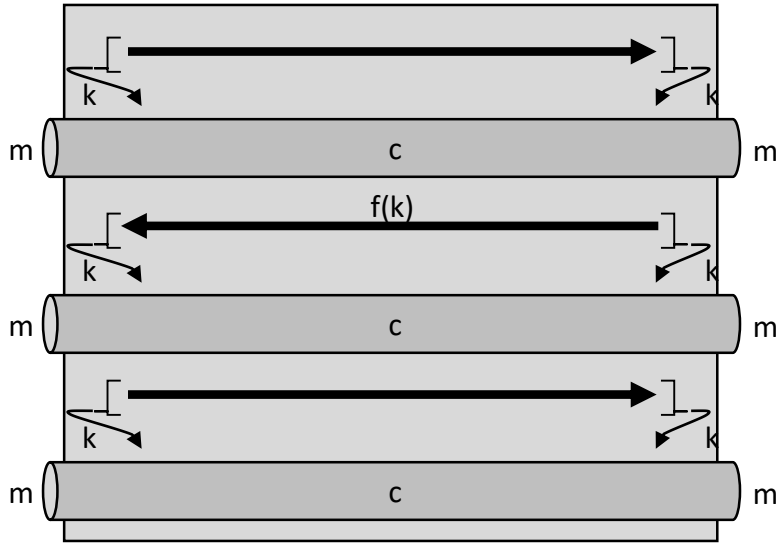
- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)

Generalization of ACCE

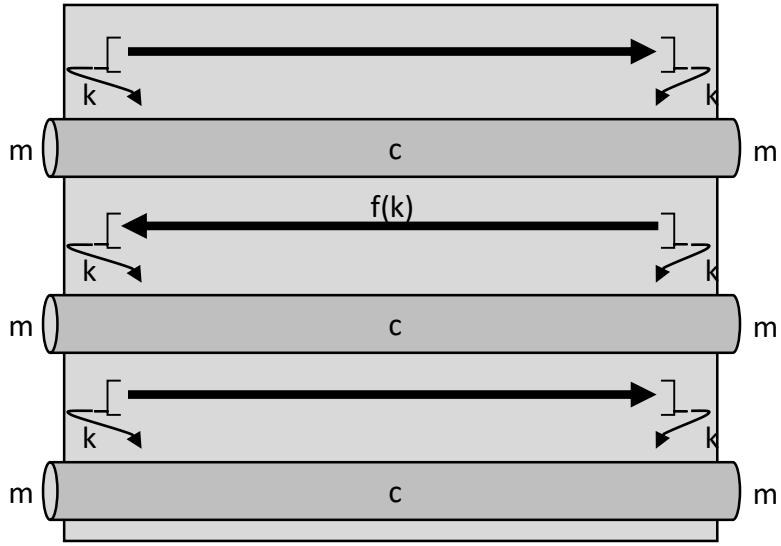
- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)
  - No pre-/post accept phase (see e.g., QUIC, TLS 1.3, Noise)

Generalization of ACCE

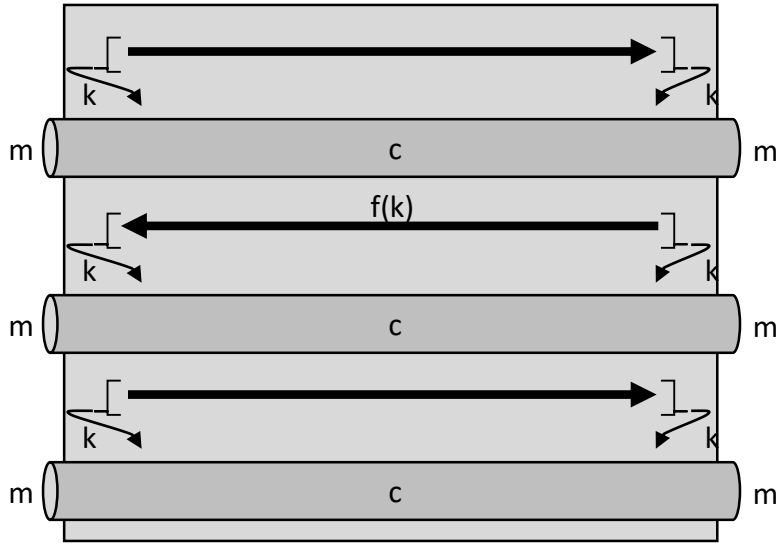
- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)
  - No pre-/post accept phase (see e.g., QUIC, TLS 1.3, Noise)
  - First ping-pong, then concurrency not mandatory (e.g., channel per stage bidirectional)

Generalization of ACCE

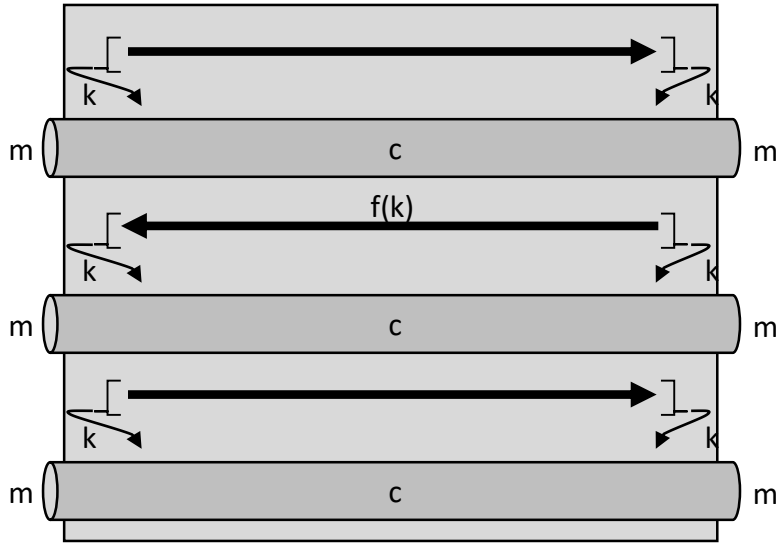
- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)
  - No pre-/post accept phase (see e.g., QUIC, TLS 1.3, Noise)
  - First ping-pong, then concurrency not mandatory (e.g., channel per stage bidirectional)
  - Length-hiding an intrinsic property?

Generalization of ACCE

- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)
  - No pre-/post accept phase (see e.g., QUIC, TLS 1.3, Noise)
  - First ping-pong, then concurrency not mandatory (e.g., channel per stage bidirectional)
  - Length-hiding an intrinsic property?
  - Initiator = client, responder = server, unilateral authentication = server authentication?

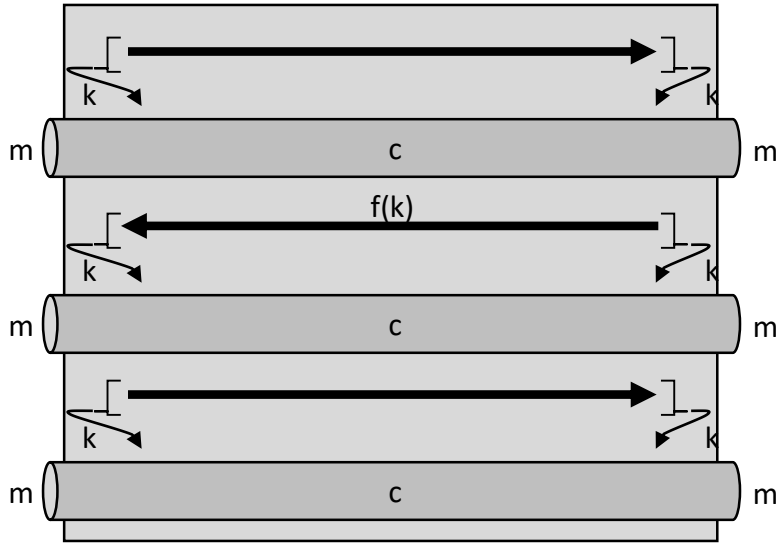
Generalization of ACCE

- ACCE modeled with TLS 1.2 in mind
- QACCE modeled with QUIC in mind
- ACCE is an own primitive

$\mathrm{KGen} \rightarrow_{\$} (sk, pk)$ $\quad$ $\mathrm{Enc}(sk, st, m, ad) \rightarrow_{\$} (st, c)$

$\mathrm{Init}(sk, pk, ad) \rightarrow_{\$} st$ $\quad$ $\mathrm{Dec}(sk, st, c, ad) \rightarrow (st, m)$

c contains whole transcript



- Generically:
  - No distinct key (e.g., suppose asymmetric PKE channels)
  - No pre-/post accept phase (see e.g., QUIC, TLS 1.3, Noise)
  - First ping-pong, then concurrency not mandatory (e.g., channel per stage bidirectional)
  - Length-hiding an intrinsic property?
  - Initiator = client, responder = server, unilateral authentication = server authentication?

Key Exchange + Channel = ?

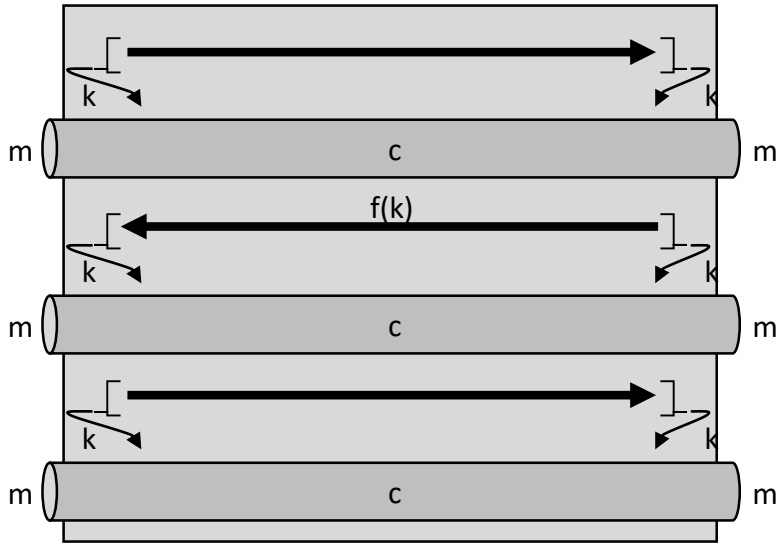
Generalization of ACCE

Modularization of ACCE

Application to Noise

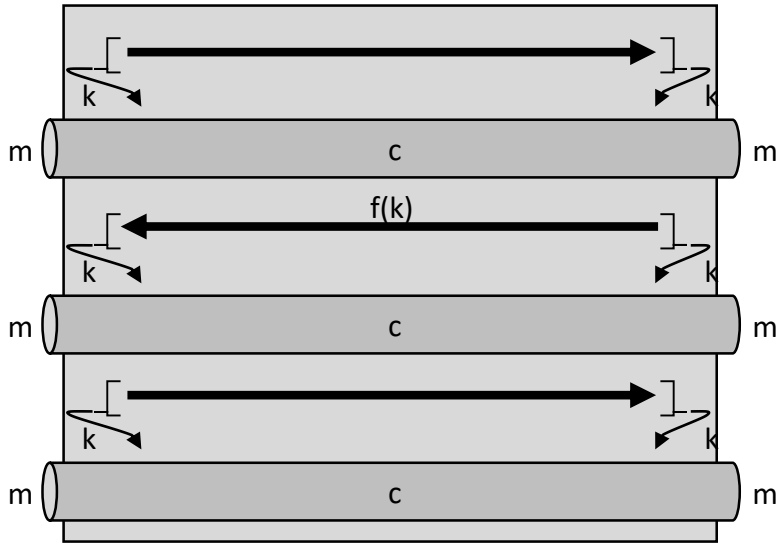
Modularization of ACCE

- Channel can provide several properties
  - Authentication
  - KCI resistance



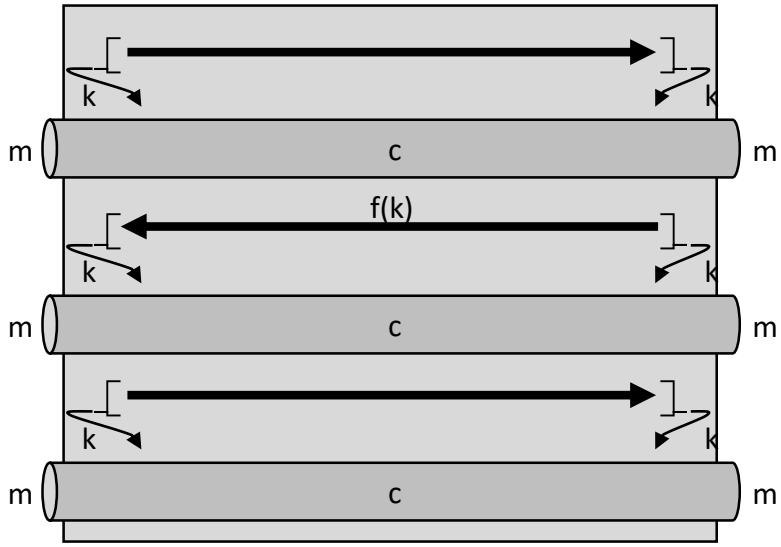
Modularization of ACCE

- Channel can provide several properties
  - Authentication
  - KCI resistance
  - Forward secrecy
  - Resistance against replay attacks



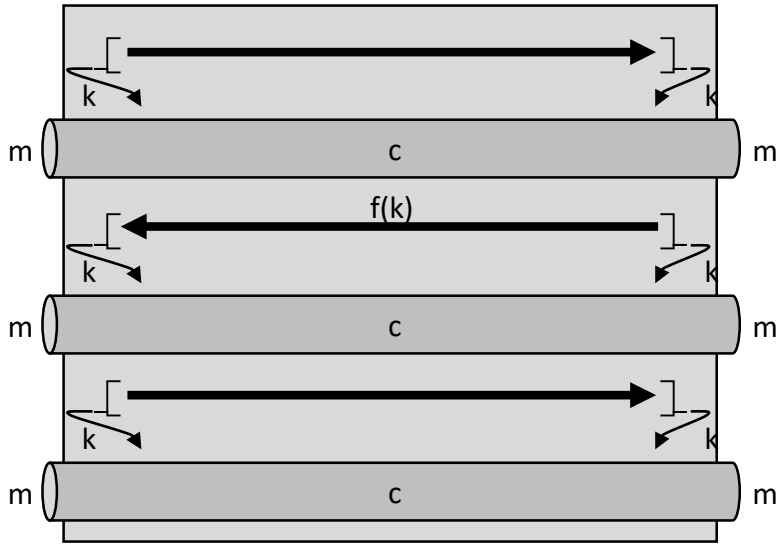
Modularization of ACCE

- Channel can provide several properties
  - Authentication
  - KCI resistance
  - Forward secrecy
  - Resistance against replay attacks
  - Resistance against weak randomness



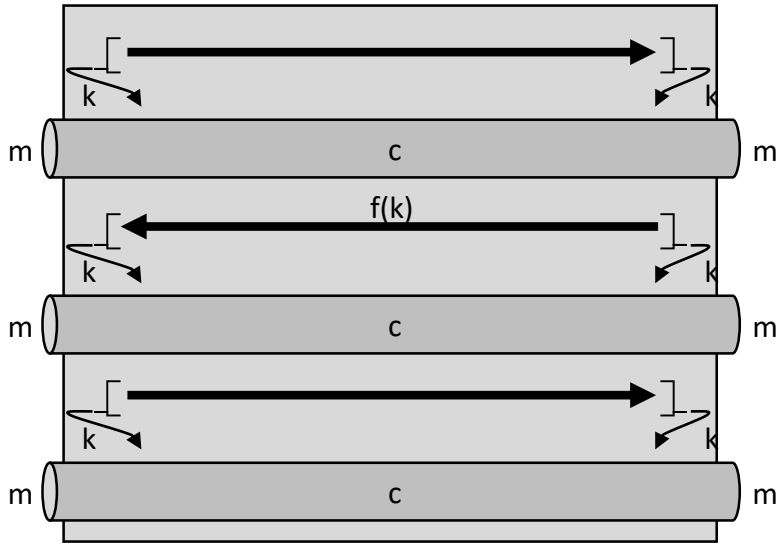
Modularization of ACCE

- Channel can provide several properties
  - Authentication
  - KCI resistance
  - Forward secrecy
  - Resistance against replay attacks
  - Resistance against weak randomness
  - We keep channel simple (i.e., stAE)



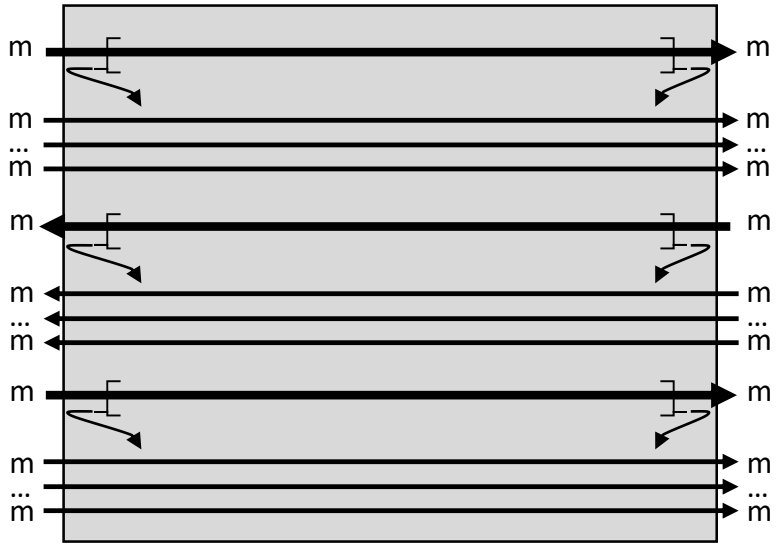
Modularization of ACCE

- Channel can provide several properties
  - Authentication
  - KCI resistance
  - Forward secrecy
  - Resistance against replay attacks
  - Resistance against weak randomness
  - We keep channel simple (i.e., stAE)
- Properties can be reached…
  - … for each party separately



Modularization of ACCE

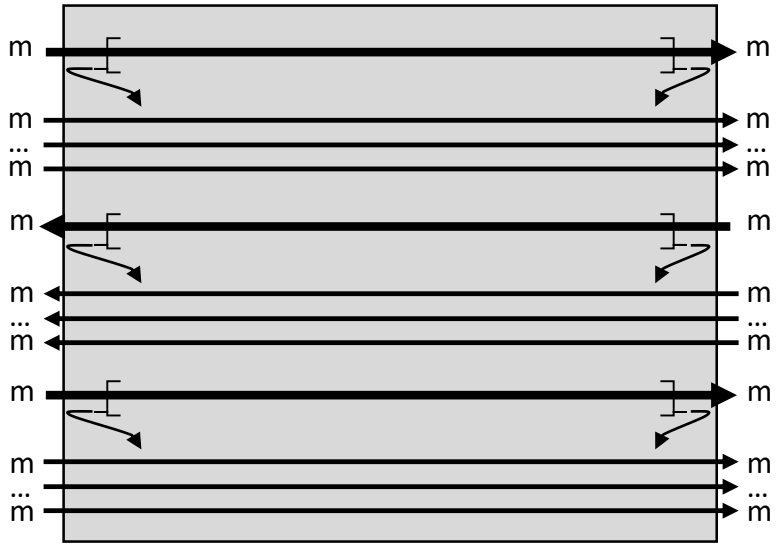
- Channel can provide several properties
  - Authentication
  - KCI resistance
  - Forward secrecy
  - Resistance against replay attacks
  - Resistance against weak randomness
  - We keep channel simple (i.e., stAE)



- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via round trips [RTs])

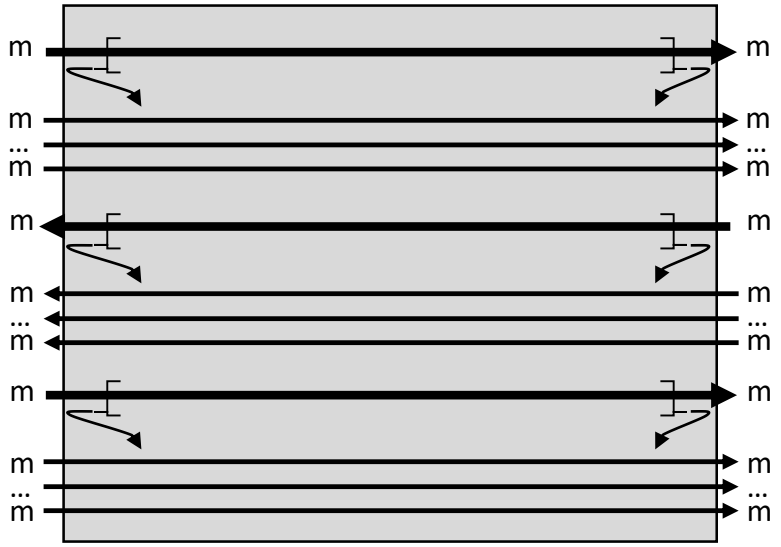
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- Round trips:



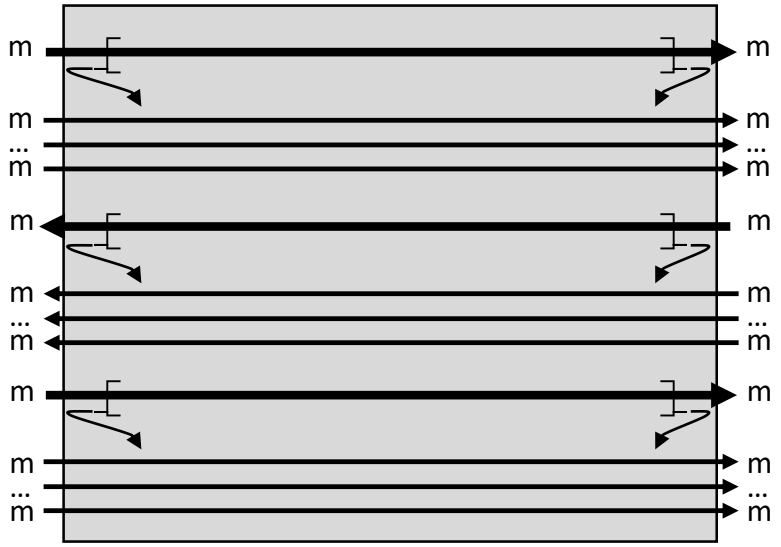
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- Round trips:
  - Interaction between parties
  - Denote *epochs* in communication



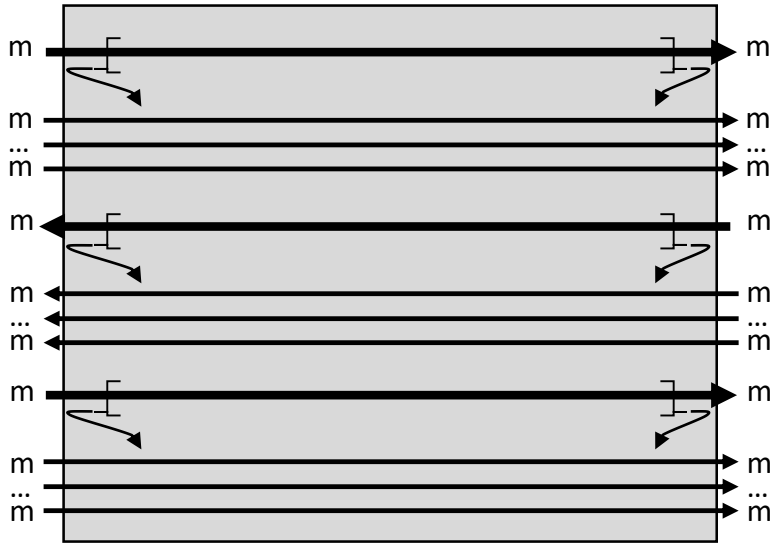
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- Round trips:
  - Interaction between parties
  - Denote *epochs* in communication
  - No *keys* to defines stages (as in MS-KE)



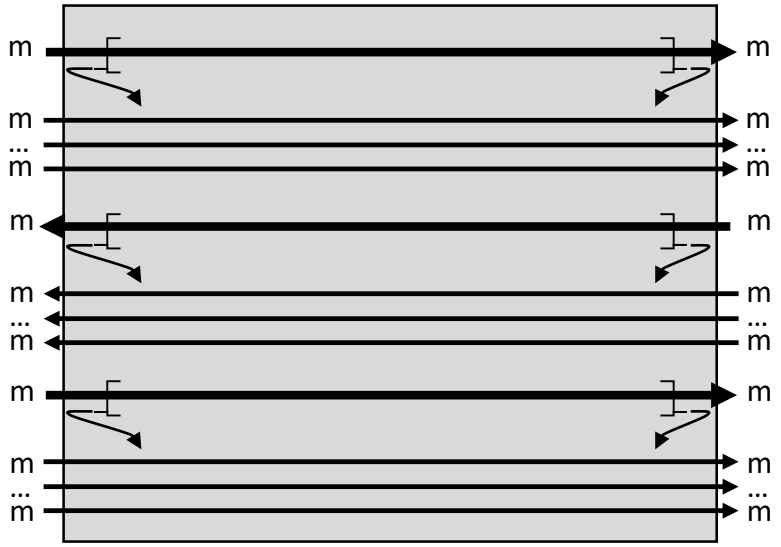
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- Round trips:
  - Interaction between parties
  - Denote *epochs* in communication
  - No *keys* to defines stages (as in MS-KE)
  - Usual in KE, ratcheting (see Signal, Bertram's talk)



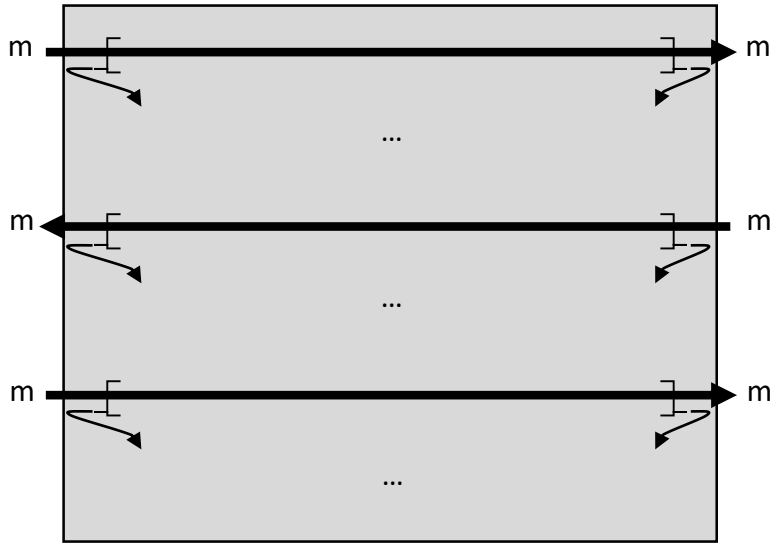
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- Round trips:
  - Interaction between parties
  - Denote *epochs* in communication
  - No *keys* to defines stages (as in MS-KE)
  - Usual in KE, ratcheting (see Signal, Bertram's talk)
- Further extension within RTs
  - Too complex for the use-case here



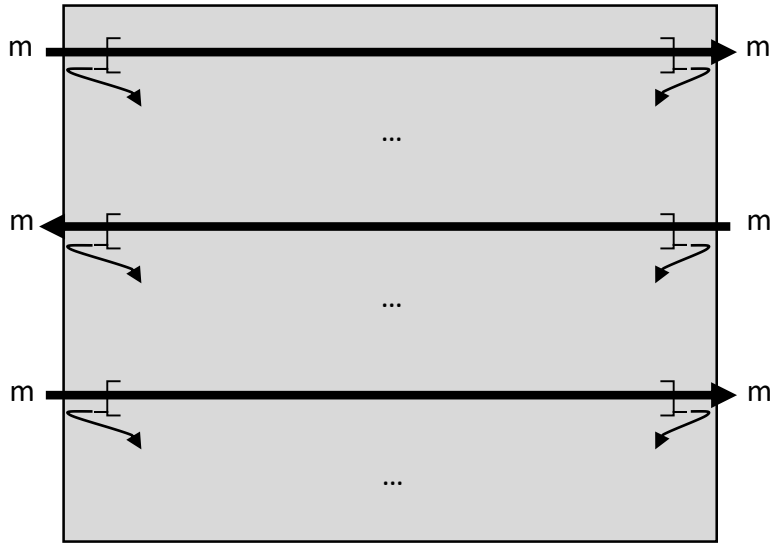
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- For each party separately:



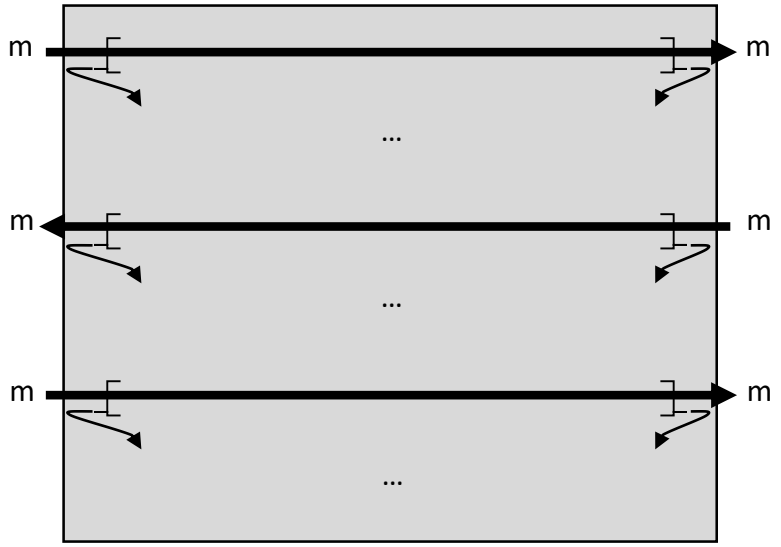
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- For each party separately:
  - Authentication *A-to-B* with message *A-to-B*



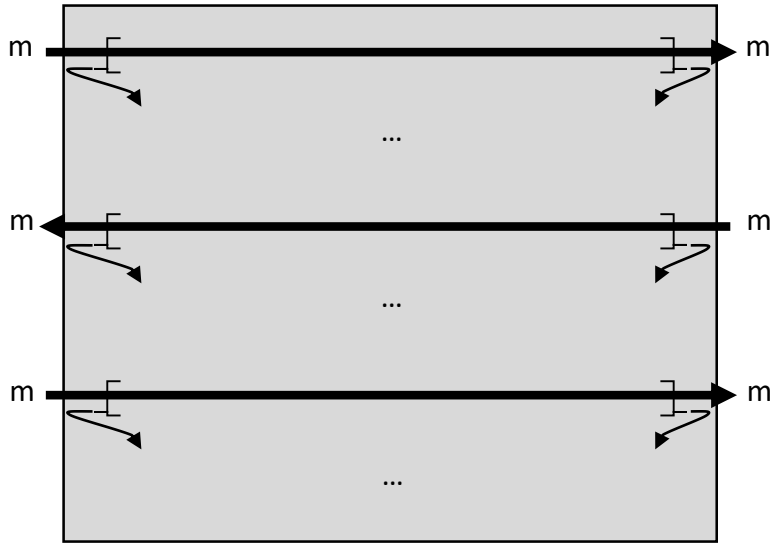
Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- For each party separately:
  - Authentication *A-to-B* with message *A-to-B*
  - E.g. resistance against weak randomness *not direction-dependent*



Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- For each party separately:
  - Authentication *A-to-B* with message *A-to-B*
  - E.g. resistance against weak randomness *not direction-dependent*

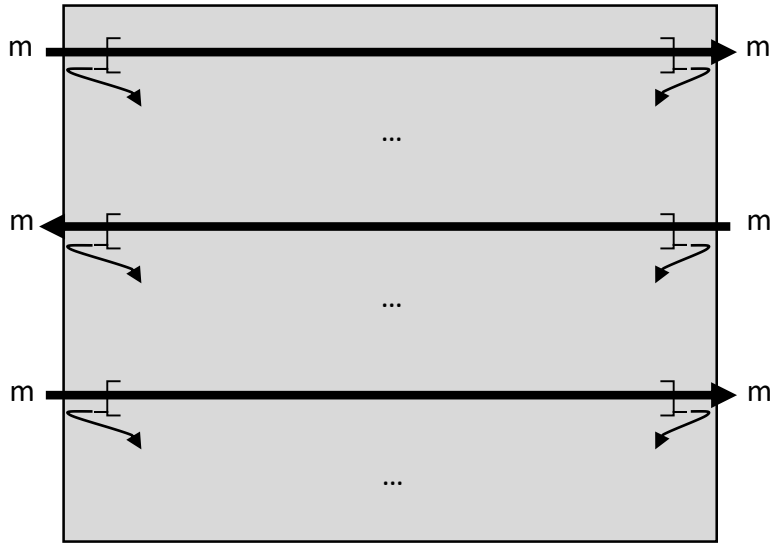


- 5*2+1 counters index our security definition:

$au^i, au^r, kc^i, kc^r, fs^i, fs^r, rp^i, rp^r, or^i, or^r, eck \in \{0, 0.5, 1, 1.5, \ldots, \infty\}$

Modularization of ACCE

- Properties can be reached…
  - … for each party separately
  - … at different *stages* during the protocol execution (via RTs)
- For each party separately:
  - Authentication *A-to-B* with message *A-to-B*
  - E.g. resistance against weak randomness *not direction-dependent*

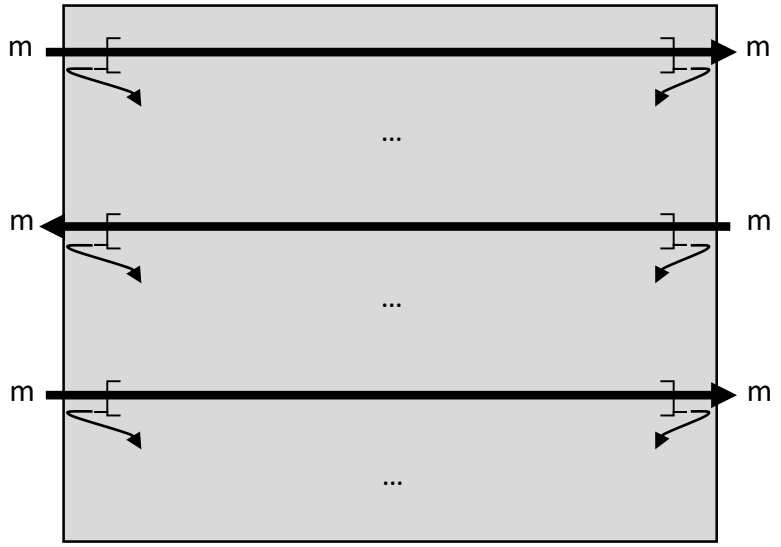


- 5*2+1 counters index our security definition:

$au^i, au^r, kc^i, kc^r, fs^i, fs^r, \mathbf{rp^i}, \mathbf{rp^r}, or^i, or^r, eck \in \{0, 0.5, 1, 1.5, \ldots, \infty\}$

Modularization of ACCE

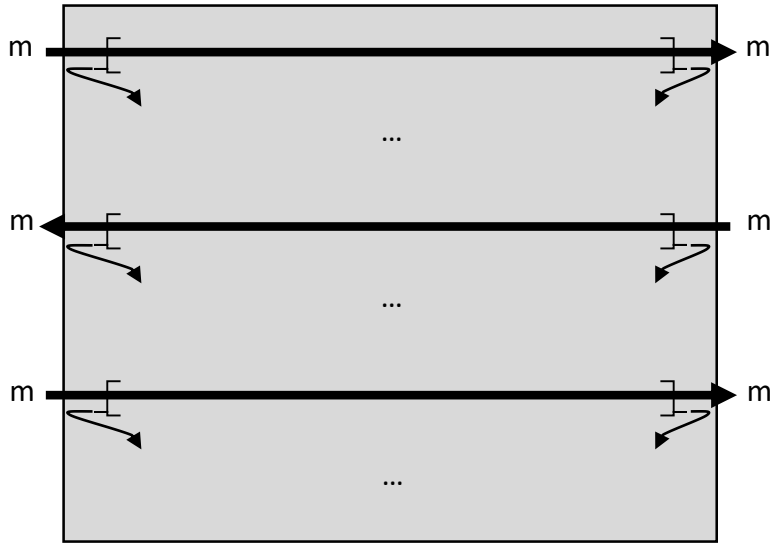
- Adversary has to guess a challenge bit
  - Enc and Dec embed challenges (stAE)



Modularization of ACCE

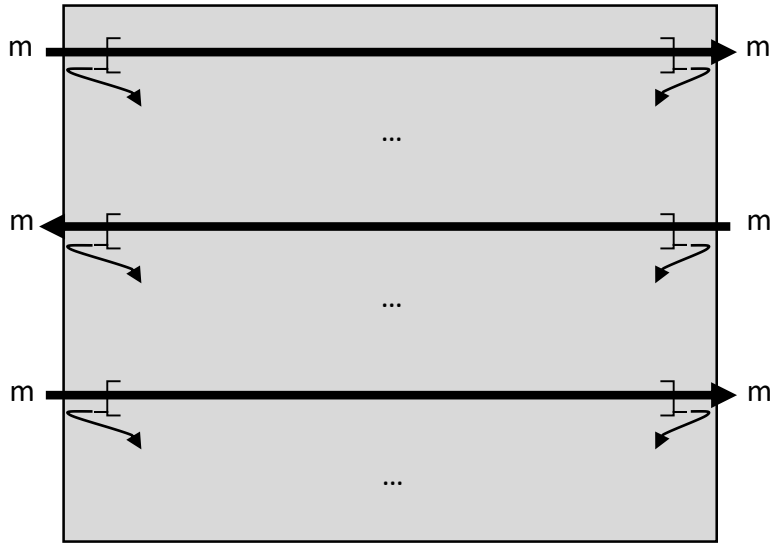
- Adversary has to guess a challenge bit
  - Enc and Dec embed challenges (stAE)
  - Adversarial behavior leaks bits of some RTs, but some must stay secure

  → Challenge bits for each RT



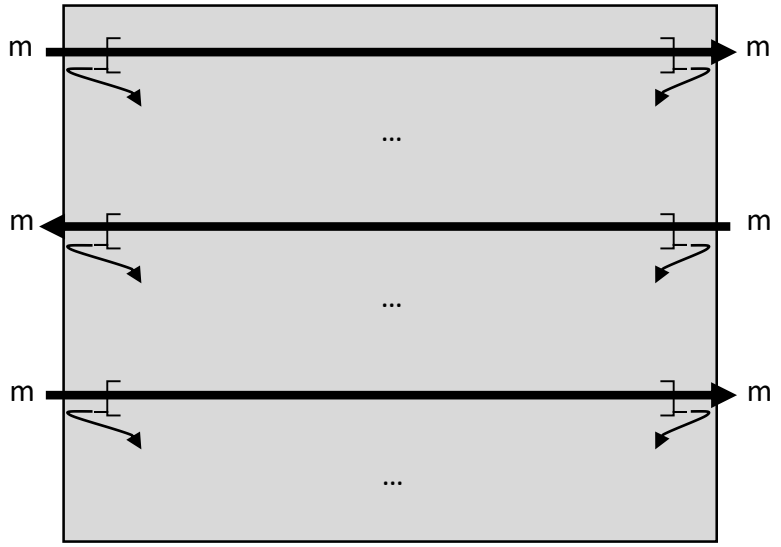
Modularization of ACCE

- Adversary can
  - Actively attack sessions
  - Corrupt parties
  - Reveal session randomness



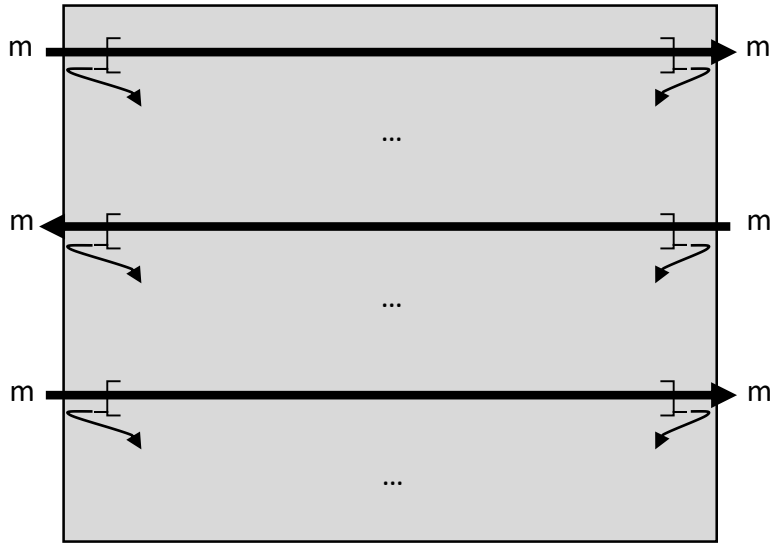
Modularization of ACCE

- Adversary can
  - Actively attack sessions
  - Corrupt parties
  - Reveal session randomness
  - Reveal session states
    - There are no keys anymore (by syntax)



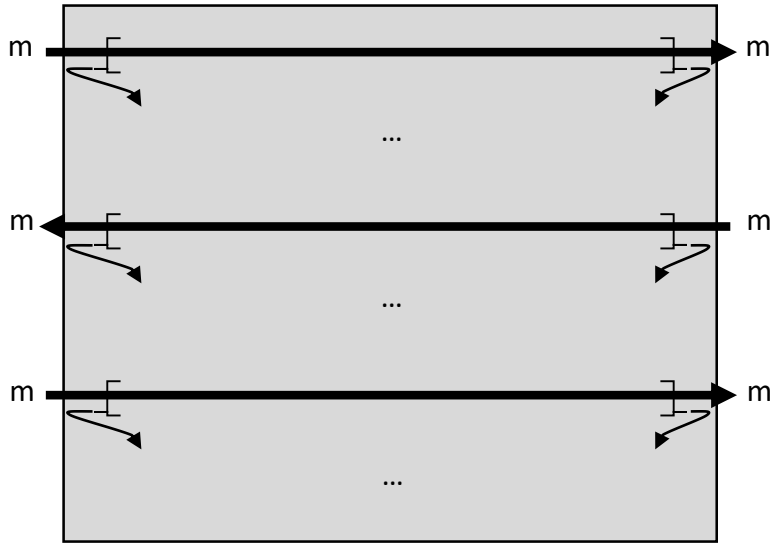
Modularization of ACCE

- Adversary can
  - Actively attack sessions
  - Corrupt parties
  - Reveal session randomness
  - Reveal session states
    - There are no keys anymore (by syntax)
    - What does **independence of sessions** mean in protocols of *long* duration (idea of Reveal in BR93)?



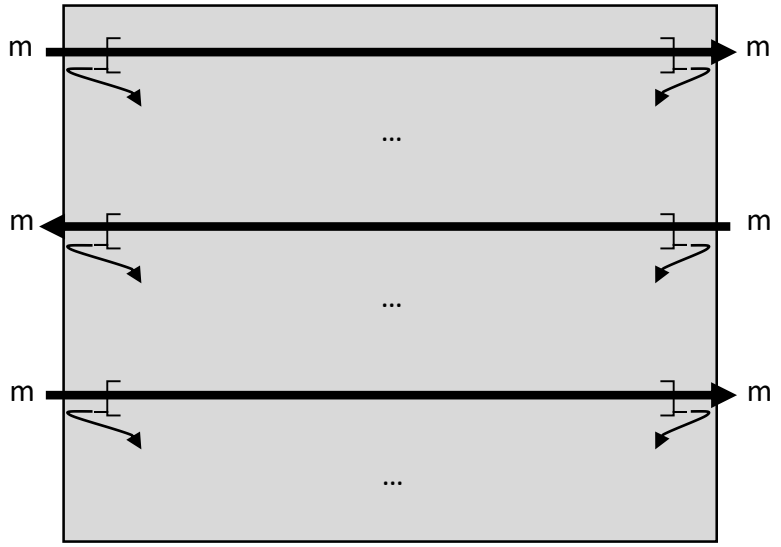
Modularization of ACCE

- Adversary can
  - Actively attack sessions
  - Corrupt parties
  - Reveal session randomness
  - Reveal session states
    - There are no keys anymore (by syntax)
    - What does **independence of sessions** mean in protocols of *long* duration (idea of Reveal in BR93)?
    - What are the effects of **replay attacks** w.r.t. session independence?



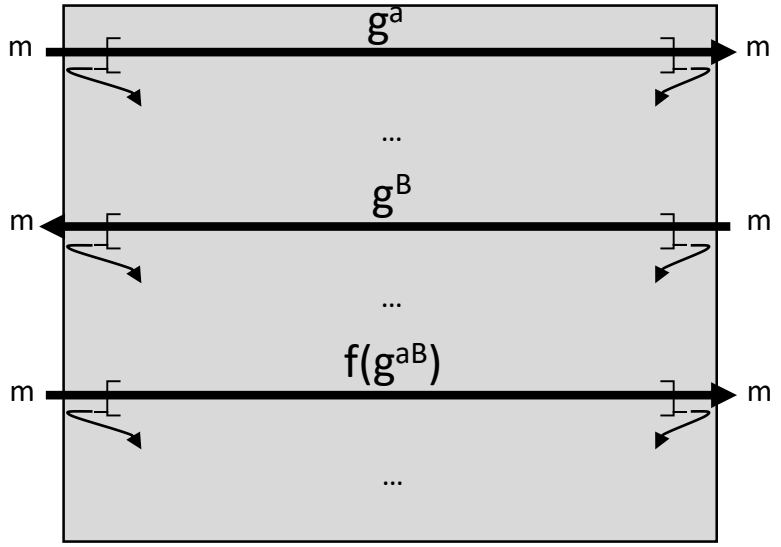
Modularization of ACCE

- Resistance against replay attacks
  - Within session modeled by stateful AE



Modularization of ACCE

- Resistance against replay attacks
  - Within session modeled by stateful AE
  - Inter session: Impact of state Reveal



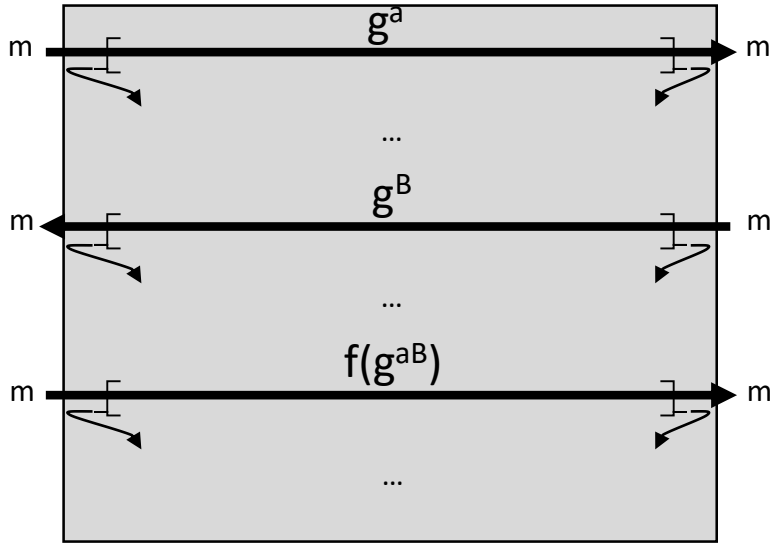
- Not only dependents on symmetric key
- Also on ephemeral asymmetric secrets

Modularization of ACCE

- Resistance against replay attacks
  - Within session modeled by stateful AE
  - Inter session: Impact of state Reveal

  rp^i, rp^r denote RT after which revealed state cannot be used to reestablish session

- Not only dependents on symmetric key
- Also on ephemeral asymmetric secrets



Key Exchange + Channel = ?

Generalization of ACCE

Modularization of ACCE

Application to Noise

Application to Noise

- Protocol framework for channel establishment
  - using DH group, AEAD, hash function, KDF
  - for different scenarios (15 patterns):

```
NN():
  -> e
  <- e, ee

NK(rs):
  <- s
  ...
  -> e, es
  <- e, ee

NX(rs):
  -> e
  <- e, ee, s, es

XN(s):
  -> e
  <- e, ee
  -> s, se

XK(s, rs):
  <- s
  ...
  -> e, es
  <- e, ee
  -> s, se

XX(s, rs):
  -> e
  <- e, ee, s, es
  -> s, se
```

```
KN(s):
  -> s
  ...
  -> e
  <- e, ee, se

KK(s, rs):
  -> s
  <- s
  ...
  -> e, es, ss
  <- e, ee, se

KX(s, rs):
  -> s
  ...
  -> e
  <- e, ee, se, s, es

IN(s):
  -> e, s
  <- e, ee, se

IK(s, rs):
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

IX(s, rs):
  -> e, s
  <- e, ee, se, s, es
```

Application to Noise

- Protocol framework for channel establishment
  - using DH group, AEAD, hash function, KDF
  - for different scenarios (15 patterns):
    - Who knows whom a priori?
    - Who should authenticate?
    - How fast should messages be transmitted?
    - Which further properties shall be reached (forward secrecy, identity hiding, …)?

```
NN():
  -> e
  <- e, ee

NK(rs):
  <- s
  ...
  -> e, es
  <- e, ee

NX(rs):
  -> e
  <- e, ee, s, es

XN(s):
  -> e
  <- e, ee
  -> s, se

XK(s, rs):
  <- s
  ...
  -> e, es
  <- e, ee
  -> s, se

XX(s, rs):
  -> e
  <- e, ee, s, es
  -> s, se
```

```
KN(s):
  -> s
  ...
  -> e
  <- e, ee, se

KK(s, rs):
  -> s
  <- s
  ...
  -> e, es, ss
  <- e, ee, se

KX(s, rs):
  -> s
  ...
  -> e
  <- e, ee, se, s, es

IN(s):
  -> e, s
  <- e, ee, se

IK(s, rs):
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

IX(s, rs):
  -> e, s
  <- e, ee, se, s, es
```

Application to Noise

- Protocol framework for channel establishment
  - using DH group, AEAD, hash function, KDF
  - for different scenarios (15 patterns):
  - implemented in Java, C, Haskell, Python, Javascript, …
  - used in WhatsApp, Wireguard, Slack, …
  - for homogenous networks
    (i.e., all parties are configured equally)

```
NN():
  -> e
  <- e, ee

NK(rs):
  <- s
  ...
  -> e, es
  <- e, ee

NX(rs):
  -> e
  <- e, ee, s, es

XN(s):
  -> e
  <- e, ee
  -> s, se

XK(s, rs):
  <- s
  ...
  -> e, es
  <- e, ee
  -> s, se

XX(s, rs):
  -> e
  <- e, ee, s, es
  -> s, se
```

```
KN(s):
  -> s
  ...
  -> e
  <- e, ee, se

KK(s, rs):
  -> s
  <- s
  ...
  -> e, es, ss
  <- e, ee, se

KX(s, rs):
  -> s
  ...
  -> e
  <- e, ee, se, s, es

IN(s):
  -> e, s
  <- e, ee, se

IK(s, rs):
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

IX(s, rs):
  -> e, s
  <- e, ee, se, s, es
```

Application to Noise

- Protocol framework for channel establishment
  - using DH group, AEAD, hash function, KDF
  - for different scenarios (15 patterns):
  - implemented in Java, C, Haskell, Python, Javascript, …
  - used in WhatsApp, Wireguard, Slack, …
  - for homogenous networks
    (i.e., all parties are configured equally)
- Security claimed but not proven yet
  - Concurrent work by Nadim Kobeissi
    (noiseexplorer.com) using ProVerif

```
NN():
  -> e
  <- e, ee

NK(rs):
  <- s
  ...
  -> e, es
  <- e, ee

NX(rs):
  -> e
  <- e, ee, s, es

XN(s):
  -> e
  <- e, ee
  -> s, se

XK(s, rs):
  <- s
  ...
  -> e, es
  <- e, ee
  -> s, se

XX(s, rs):
  -> e
  <- e, ee, s, es
  -> s, se
```

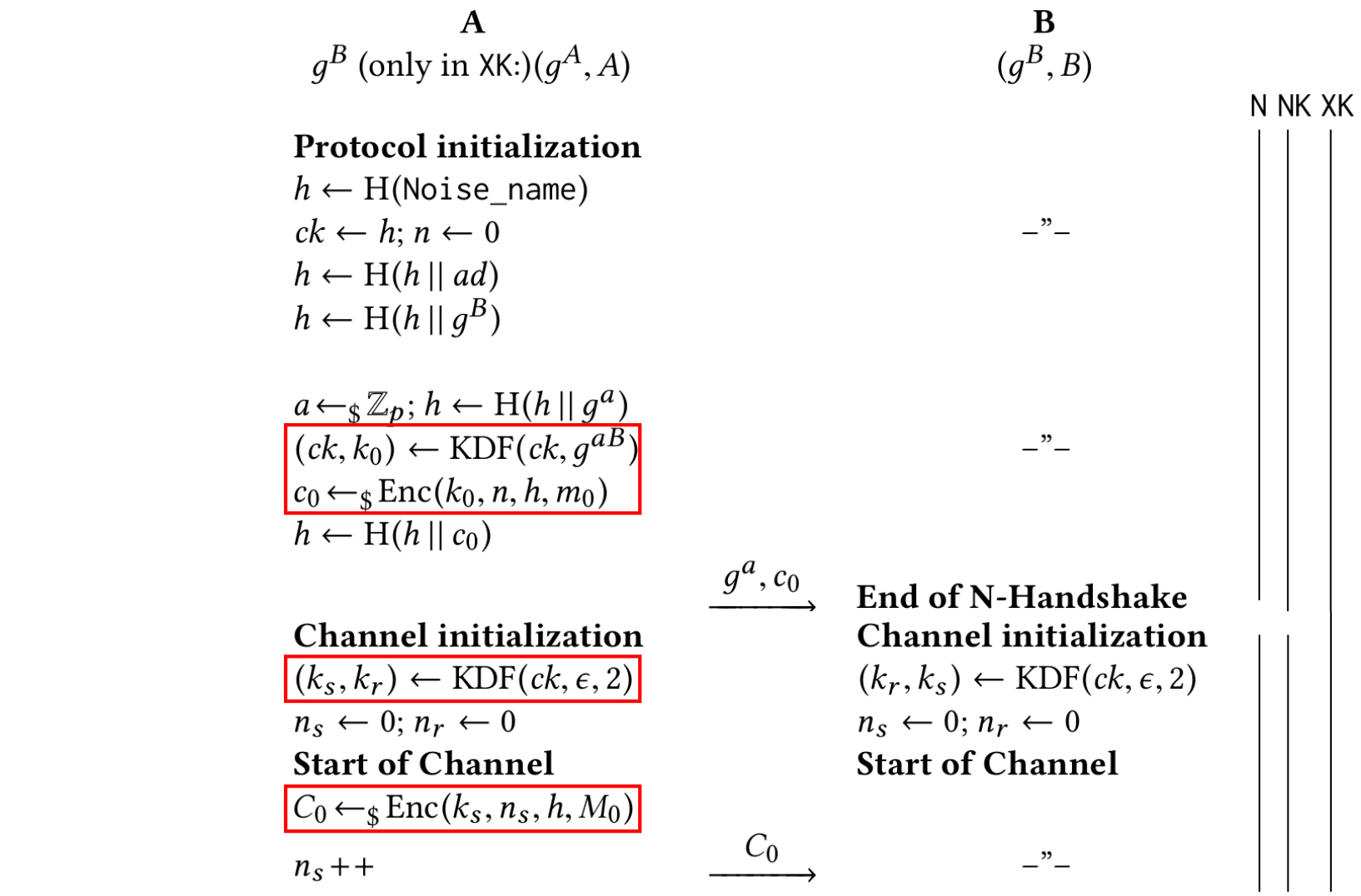
```
KN(s):
  -> s
  ...
  -> e
  <- e, ee, se

KK(s, rs):
  -> s
  <- s
  ...
  -> e, es, ss
  <- e, ee, se

KX(s, rs):
  -> s
  ...
  -> e
  <- e, ee, se, s, es

IN(s):
  -> e, s
  <- e, ee, se

IK(s, rs):
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

IX(s, rs):
  -> e, s
  <- e, ee, se, s, es
```

Application to Noise

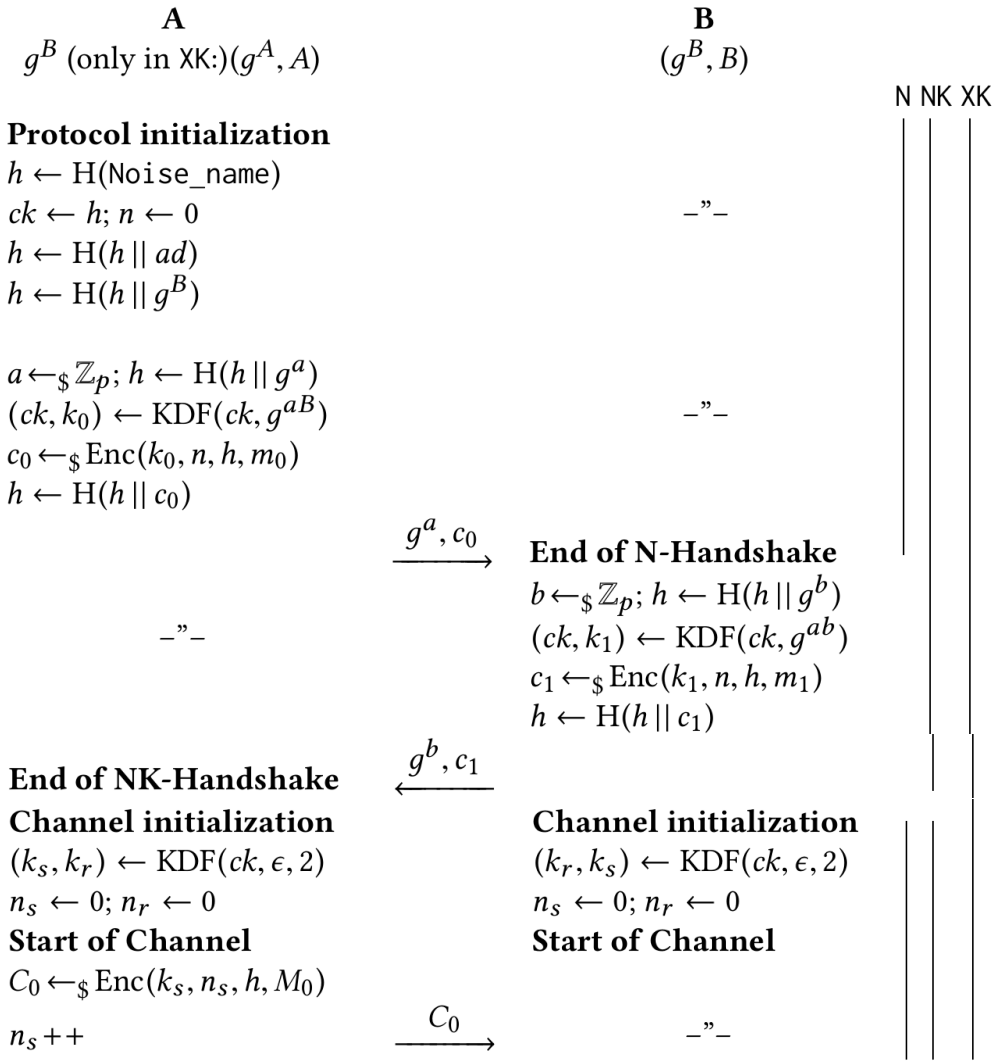
A		B
g^B (only in XK:)(g^A, A)		(g^B, B)
Protocol initialization		
$h \leftarrow \mathrm{H}(\texttt{Noise\_name})$		
$ck \leftarrow h; n \leftarrow 0$		–"–
$h \leftarrow \mathrm{H}(h \,\|\|\, ad)$		
$h \leftarrow \mathrm{H}(h \,\|\|\, g^B)$		
$a \leftarrow_\$ \mathbb{Z}_p; h \leftarrow \mathrm{H}(h \,\|\|\, g^a)$		
$(ck, k_0) \leftarrow \mathrm{KDF}(ck, g^{aB})$		–"–
$c_0 \leftarrow_\$ \mathrm{Enc}(k_0, n, h, m_0)$		
$h \leftarrow \mathrm{H}(h \,\|\, c_0)$		
	$\xrightarrow{g^a, c_0}$	**End of N-Handshake**
Channel initialization		**Channel initialization**
$(k_s, k_r) \leftarrow \mathrm{KDF}(ck, \epsilon, 2)$		$(k_r, k_s) \leftarrow \mathrm{KDF}(ck, \epsilon, 2)$
$n_s \leftarrow 0; n_r \leftarrow 0$		$n_s \leftarrow 0; n_r \leftarrow 0$
Start of Channel		**Start of Channel**
$C_0 \leftarrow_\$ \mathrm{Enc}(k_s, n_s, h, M_0)$		
$n_s{+}{+}$	$\xrightarrow{C_0}$	–"–

N NK XK

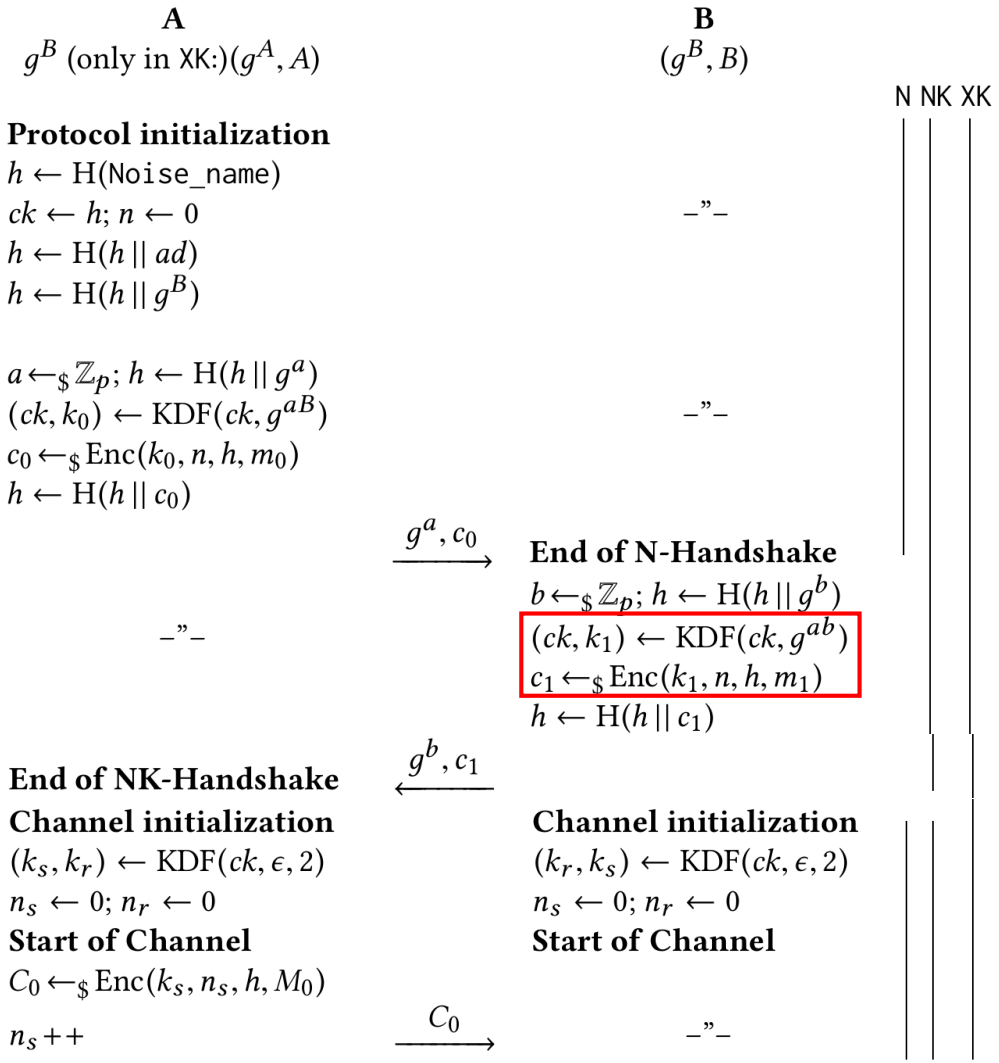
Application to Noise



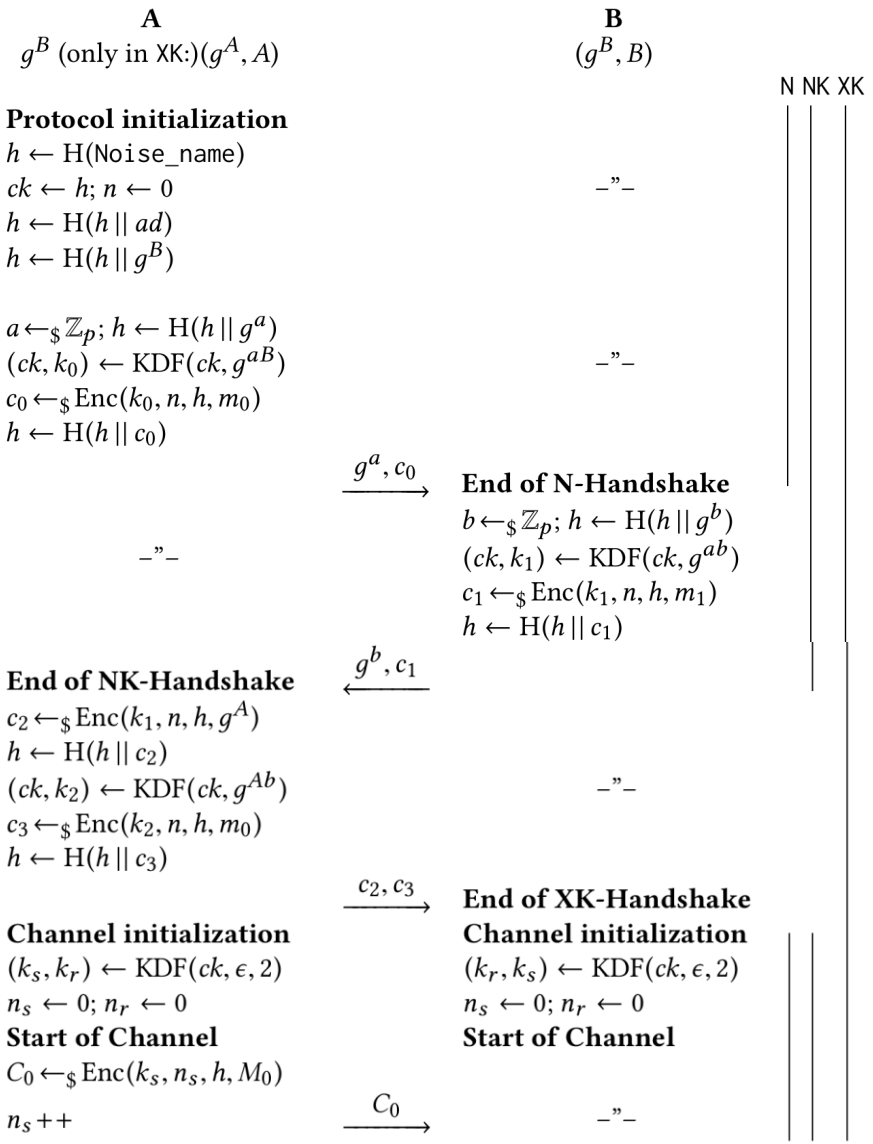
Application to Noise



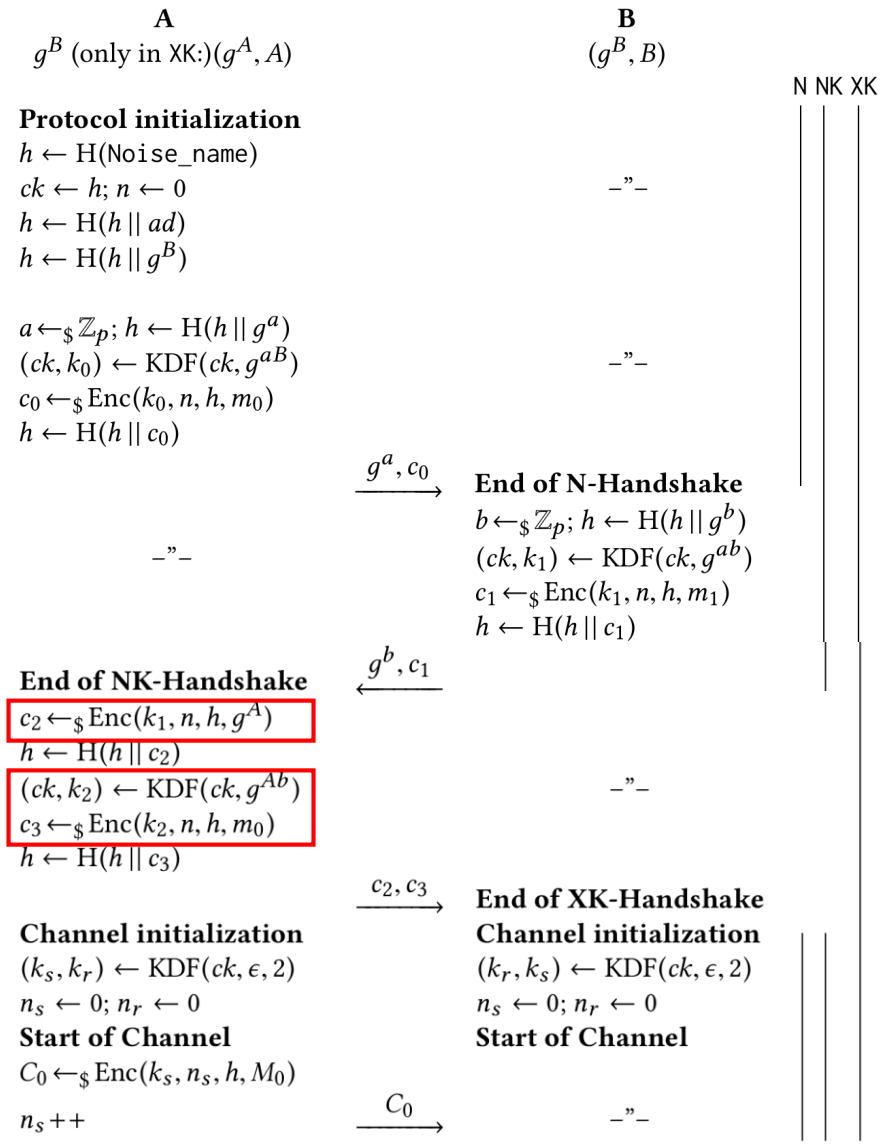
Application to Noise



Application to Noise



Application to Noise



Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance

The authentication properties are:

0. **No authentication.** This payload may have been sent by any party, including an active attacker.

1. **Sender authentication *vulnerable* to key-compromise impersonation (KCI).** The sender authentication is based on a static-static DH (`"ss"`) involving both parties' static key pairs. If the recipient's long-term private key has been compromised, this authentication can be forged. Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.

2. **Sender authentication *resistant* to key-compromise impersonation (KCI).** The sender authentication is based on an ephemeral-static DH (`"es"` or `"se"`) between the sender's static key pair and the recipient's ephemeral key pair. Assuming the corresponding private keys are secure, this authentication cannot be forged.

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks

The authentication properties are:

0. **No authentication.** This payload may have been sent by any party, including an active attacker.
1. **Sender authentication *vulnerable* to key-compromise impersonation (KCI).** The sender authentication is based on a static-static DH (`"ss"`) involving both parties' static key pairs. If the recipient's long-term private key has been compromised, this authentication can be forged. Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
2. **Sender authentication *resistant* to key-compromise impersonation (KCI).** The sender authentication is based on an ephemeral-static DH (`"es"` or `"se"`) between the sender's static key pair and the recipient's ephemeral key pair. Assuming the corresponding private keys are secure, this authentication cannot be forged.

The confidentiality properties are:

0. **No confidentiality.** This payload is sent in cleartext.
1. **Encryption to an ephemeral recipient.** This payload has forward secrecy, since encryption involves an ephemeral-ephemeral DH (`"ee"`). However, the sender has not authenticated the recipient, so this payload might be sent to any party, including an active attacker.
2. **Encryption to a known recipient, forward secrecy for sender compromise only, vulnerable to replay.** This payload is encrypted based only on DHs involving the recipient's static key pair. If the recipient's static private key is compromised, even at a later date, this payload can be decrypted. This message can also be replayed, since there's no ephemeral contribution from the recipient.
3. **Encryption to a known recipient, weak forward secrecy.** This payload is encrypted based on an ephemeral-ephemeral DH and also an ephemeral-static DH involving the recipient's static key pair. However, the binding between the recipient's alleged ephemeral public key and the recipient's static public key hasn't been verified by the sender, so the recipient's alleged ephemeral public key may have been forged by an active attacker. In this case, the attacker could later compromise the recipient's static private key to decrypt the payload. Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
4. **Encryption to a known recipient, weak forward secrecy if the sender's private key has been compromised.** This payload is encrypted based on an ephemeral-ephemeral DH, and also based on an ephemeral-static DH involving the recipient's static key pair. However, the binding between the recipient's alleged ephemeral public and the recipient's static public key has only been verified based on DHs involving both those public keys and the sender's static private key. Thus, if the sender's static private key was previously compromised, the recipient's alleged ephemeral public key may have been forged by an active attacker. In this case, the attacker could later compromise the intended recipient's static private key to decrypt the payload (this is a variant of a "KCI" attack enabling a "weak forward secrecy" attack). Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
5. **Encryption to a known recipient, strong forward secrecy.** This payload is encrypted based on an ephemeral-ephemeral DH as well as an ephemeral-static DH with the recipient's static key pair. Assuming the ephemeral private keys are secure, and the recipient is not being actively impersonated by an attacker that has stolen its static private key, this payload cannot be decrypted.

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks

The confidentiality properties are:

0. **No confidentiality.** This payload is sent in cleartext.
1. **Encryption to an ephemeral recipient.** This payload has forward secrecy, since encryption involves an ephemeral-ephemeral DH (`"ee"`). However, the sender has not authenticated the recipient, so this payload might be sent to any party, including an active attacker.
2. **Encryption to a known recipient, forward secrecy for sender compromise only, vulnerable to replay.** This payload is encrypted based only on DHs involving the recipient's static key pair. If the recipient's static private key is compromised, even at a later date, this payload can be decrypted. This message can also be replayed, since there's no ephemeral contribution from the recipient.
3. **Encryption to a known recipient, weak forward secrecy.** This payload is encrypted based on an ephemeral-ephemeral DH and also an ephemeral-static DH involving the recipient's static key pair. However, the binding between the recipient's alleged ephemeral public key and the recipient's static public key hasn't been verified by the sender, so the recipient's alleged ephemeral public key may have been forged by an active attacker. In this case, the attacker could later compromise the recipient's static private key to decrypt the payload. Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
4. **Encryption to a known recipient, weak forward secrecy if the sender's private key has been compromised.** This payload is encrypted based on an ephemeral-ephemeral DH, and also based on an ephemeral-static DH involving the recipient's static key pair. However, the binding between the recipient's alleged ephemeral public and the recipient's static public key has only been verified based on DHs involving both those public keys and the sender's static private key. Thus, if the sender's static private key was previously compromised, the recipient's alleged ephemeral public key may have been forged by an active attacker. In this case, the attacker could later compromise the intended recipient's static private key to decrypt the payload (this is a variant of a "KCI" attack enabling a "weak forward secrecy" attack). Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
5. **Encryption to a known recipient, strong forward secrecy.** This payload is encrypted based on an ephemeral-ephemeral DH as well as an ephemeral-static DH with the recipient's static key pair. Assuming the ephemeral private keys are secure, and the recipient is not being actively impersonated by an attacker that has stolen its static private key, this payload cannot be decrypted.

The authentication properties are:

0. **No authentication.** This payload may have been sent by any party, including an active attacker.
1. **Sender authentication *vulnerable* to key-compromise impersonation (KCI).** The sender authentication is based on a static-static DH (`"ss"`) involving both parties' static key pairs. If the recipient's long-term private key has been compromised, this authentication can be forged. Note that a future version of Noise might include signatures, which could improve this security property, but brings other trade-offs.
2. **Sender authentication *resistant* to key-compromise impersonation (KCI).** The sender authentication is based on an ephemeral-static DH (`"es"` or `"se"`) between the sender's static key pair and the recipient's ephemeral key pair. Assuming the corresponding private keys are secure, this authentication cannot be forged.

	Authentication	Confidentiality
N	0	2
K	1	2
X	1	2
NN -> e <- e, ee ->	0 0 0	0 1 1
NK <- s ... -> e, es <- e, ee ->	 0 2 0	 2 1 5
NX -> e <- e, ee, s, es ->	 0 2 0	 0 1 5
XN -> e <- e, ee -> s, se <-	 0 0 2 0	 0 1 1 5
XK <- s ... -> e, es <- e, ee -> s, se <-	 0 2 2 2	 2 1 5 5

XX -> e <- e, ee, s, es -> s, se <-	 0 2 2	 0 1 5 5
KN -> s ... -> e <- e, ee, se -> <-	 0 0 2 0	 0 3 1 5
KK -> s <- s ... -> e, es, ss <- e, ee, se -> <-	 1 2 2 2	 2 4 5 5
KX -> s ... -> e <- e, ee, se, s, es -> <-	 0 2 2 2	 0 3 5 5
IN -> e, s <- e, ee, se -> <-	 0 0 2 0	 0 3 1 5
IK <- s ... -> e, es, s, ss <- e, ee, se -> <-	 1 2 2 2	 2 4 5 5
IX -> e, s <- e, ee, se, s, es -> <-	 0 2 2 2	 0 3 5 5

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks

	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞
KN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
KK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
KX	1.5	1	1.5	1	0	0	1	0	1	1	∞
IN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
IK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
IX	1.5	1	1.5	1	0	0	1	0	1	1	∞

Table 1: Counters for theorems. au^ρ, kc^ρ were extracted from specification [25]. fs^ρ, rp^ρ are related to their definition in the specification (but adapted to our model). or^ρ, eck defined purely with respect to the model. We give proofs for the patterns marked with a *.

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks

	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞
KN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
KK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
KX	1.5	1	1.5	1	0	0	1	0	1	1	∞
IN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
IK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
IX	1.5	1	1.5	1	0	0	1	0	1	1	∞

Table 1: Counters for theorems. au^ρ, kc^ρ were extracted from specification [25]. fs^ρ, rp^ρ are related to their definition in the specification (but adapted to our model). or^ρ, eck defined purely with respect to the model. We give proofs for the patterns marked with a *.

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy
    + Resistance against replay attacks

	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞
KN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
KK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
KX	1.5	1	1.5	1	0	0	1	0	1	1	∞
IN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
IK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
IX	1.5	1	1.5	1	0	0	1	0	1	1	∞

Table 1: Counters for theorems. au^ρ, kc^ρ were extracted from specification [25]. fs^ρ, rp^ρ are related to their definition in the specification (but adapted to our model). or^ρ, eck defined purely with respect to the model. We give proofs for the patterns marked with a *.

Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks

	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞
KN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
KK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
KX	1.5	1	1.5	1	0	0	1	0	1	1	∞
IN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
IK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
IX	1.5	1	1.5	1	0	0	1	0	1	1	∞

Table 1: Counters for theorems. au^ρ, kc^ρ were extracted from specification [25]. fs^ρ, rp^ρ are related to their definition in the specification (but adapted to our model). or^ρ, eck defined purely with respect to the model. We give proofs for the patterns marked with a *.

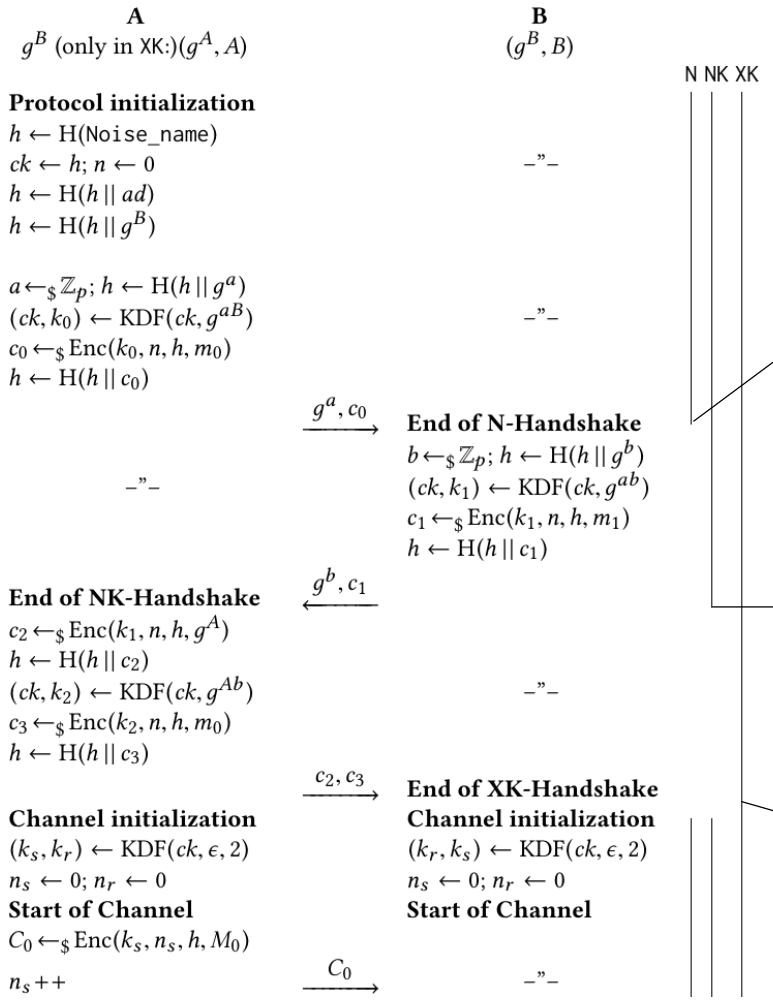
Application to Noise

- Security claimed but not proven yet
  - Authentication + KCI resistance
  - Confidentiality + Forward secrecy + Resistance against replay attacks
  - Resistance against weak randomness

	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞
KN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
KK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
KX	1.5	1	1.5	1	0	0	1	0	1	1	∞
IN	1.5	∞	1.5	∞	0	0	1	0	1	∞	∞
IK	0.5	1	1.5	1	0	1	1	1	0.5	0	0.5
IX	1.5	1	1.5	1	0	0	1	0	1	1	∞

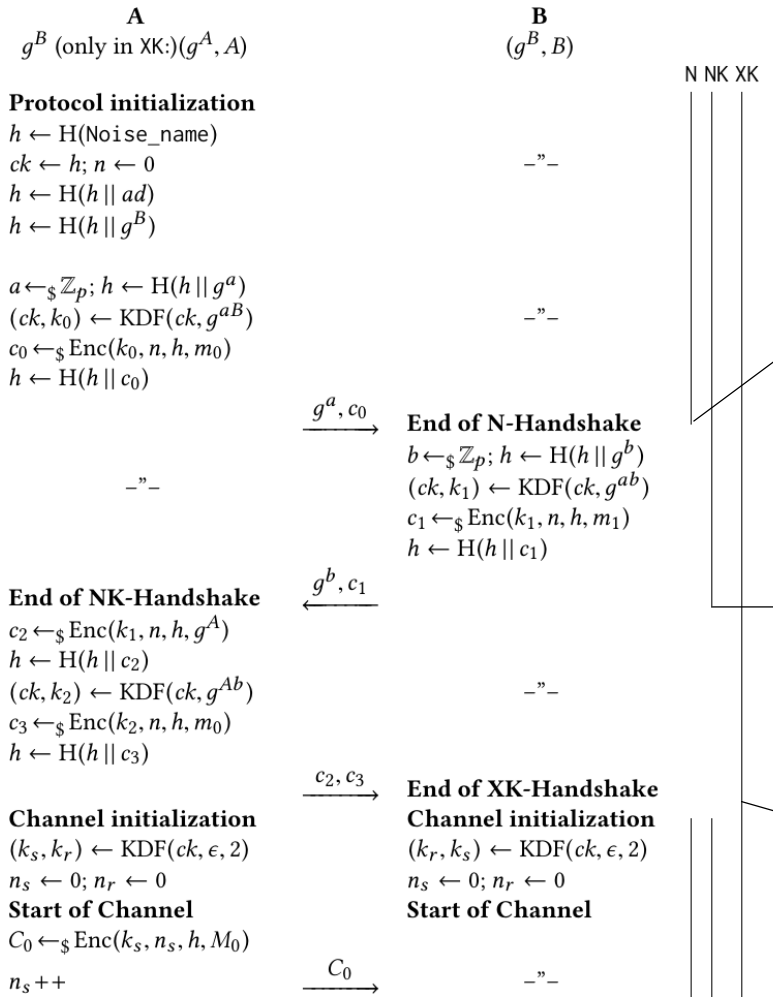
Table 1: Counters for theorems. au^ρ, kc^ρ were extracted from specification [25]. fs^ρ, rp^ρ are related to their definition in the specification (but adapted to our model). or^ρ, eck defined purely with respect to the model. We give proofs for the patterns marked with a *.

Application to Noise



	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞

Application to Noise



	au^i	au^r	kc^i	kc^r	fs^i	fs^r	rp^i	rp^r	or^i	or^r	eck
N*	∞	∞	∞	∞	0	∞	∞	∞	∞	0	∞
X*	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
K	0.5	∞	∞	∞	0	∞	∞	∞	0.5	0	∞
NN*	∞	∞	∞	∞	0	0	1	0	∞	∞	∞
NK*	∞	1	∞	1	0	1	1	1	∞	0	∞
NX*	∞	1	∞	1	0	0	1	0	∞	1	∞
XN*	1.5	∞	1.5	∞	0	0	1	0	1.5	∞	∞
XK*	1.5	1	1.5	1	0	1	1	1	1.5	0	∞
XX*	1.5	1	1.5	1	0	0	1	0	1.5	2	∞

Outlook

- Generalization of ACCE
- Modularization of ACCE
  (as MS-KE modularizes BR93)
- Computational security proofs for Noise

Outlook

- Generalization of ACCE
- Modularization of ACCE
  (as MS-KE modularizes BR93)
- Computational security proofs for Noise

- Further extensions regarding
  - Intra-epoch properties
  - Channel properties

Outlook

- Generalization of ACCE
- Modularization of ACCE
  (as MS-KE modularizes BR93)
- Computational security proofs for Noise

- Further extensions regarding
  - Intra-epoch properties
  - Channel properties

- Further properties of Noise
  - Negotiation
  - Identity hiding

Outlook

- Generalization of ACCE
- Modularization of ACCE (as MS-KE modularizes BR93)
- Computational security proofs for Noise

- Further extensions regarding
  - Intra-epoch properties
  - Channel properties

- Further properties of Noise
  - Negotiation
  - Identity hiding

- Discussions
  - What means sessions are independent in protocols of *long* duration?
  - Is ACCE as *bad* as it is advertised?
  - What can MS-KE learn from our model?
  - Can abstract (MS-)KE with channel in which key is used to a higher level?