

# Cryptographic Reverse Firewalls

Ilya Mironov (Google)  
Noah Stephens-Davidowitz (NYU)

(Work done at Microsoft Research)

# Classical Crypto

# Classical Crypto



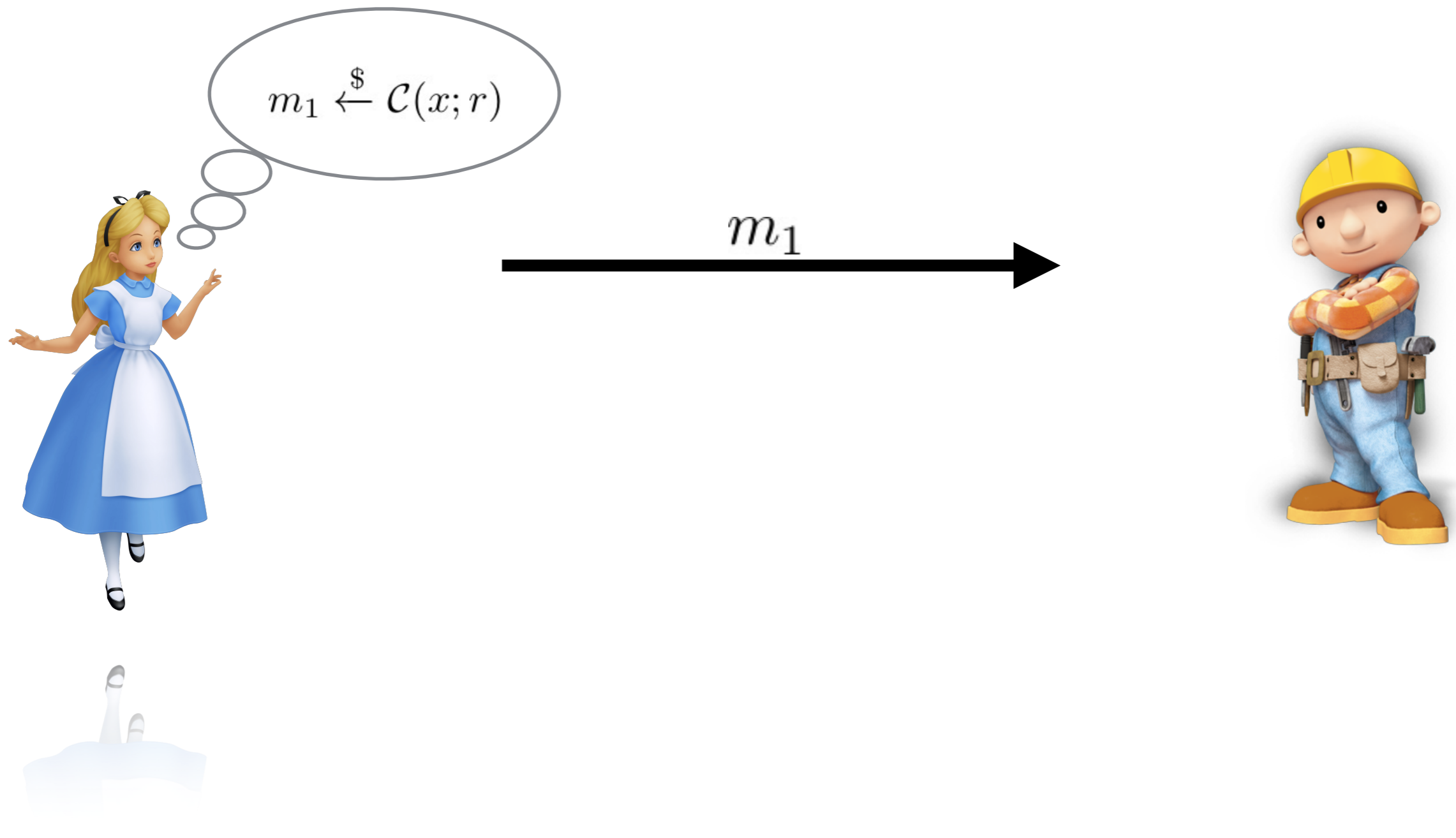
# Classical Crypto



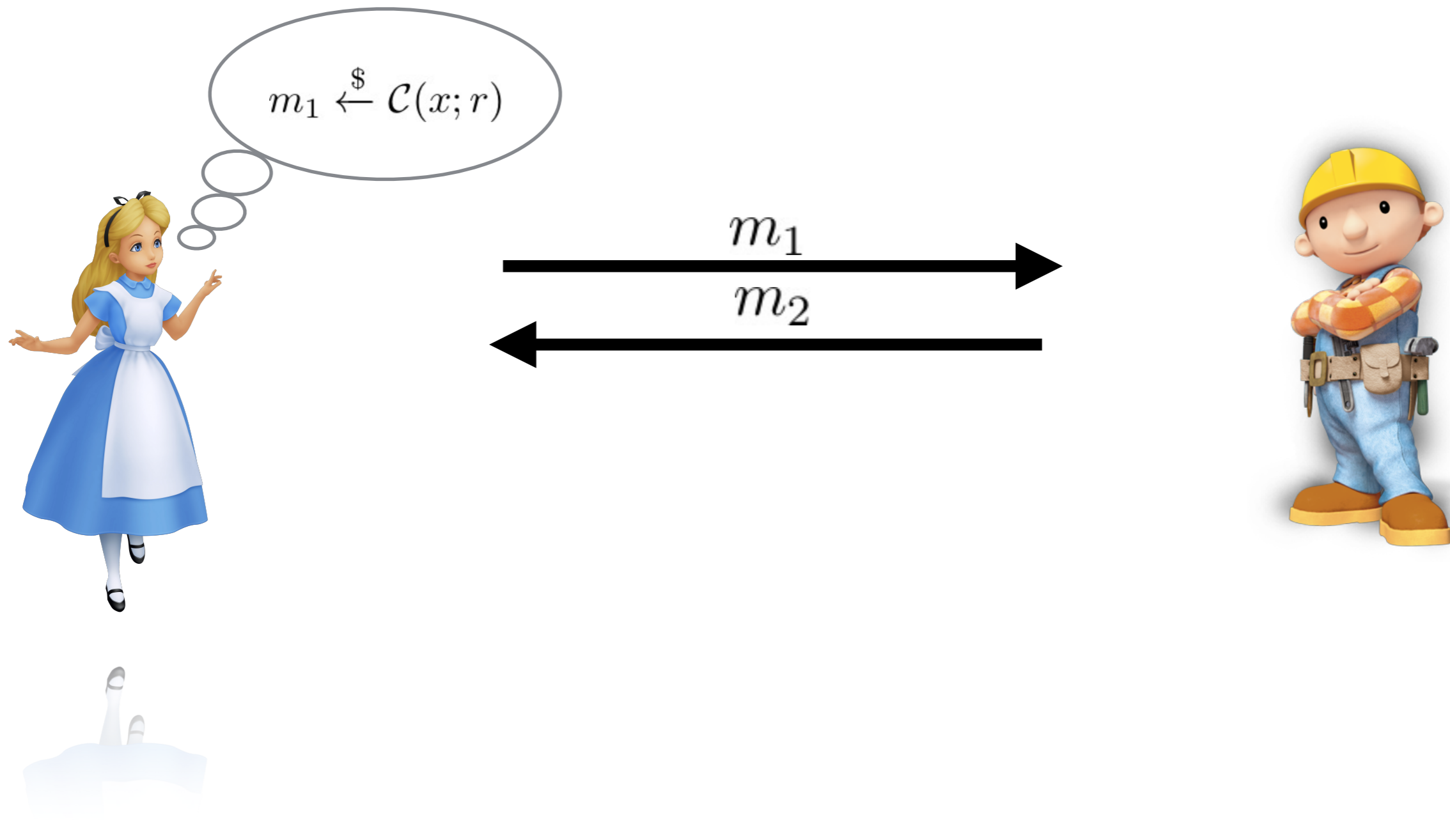
# Classical Crypto



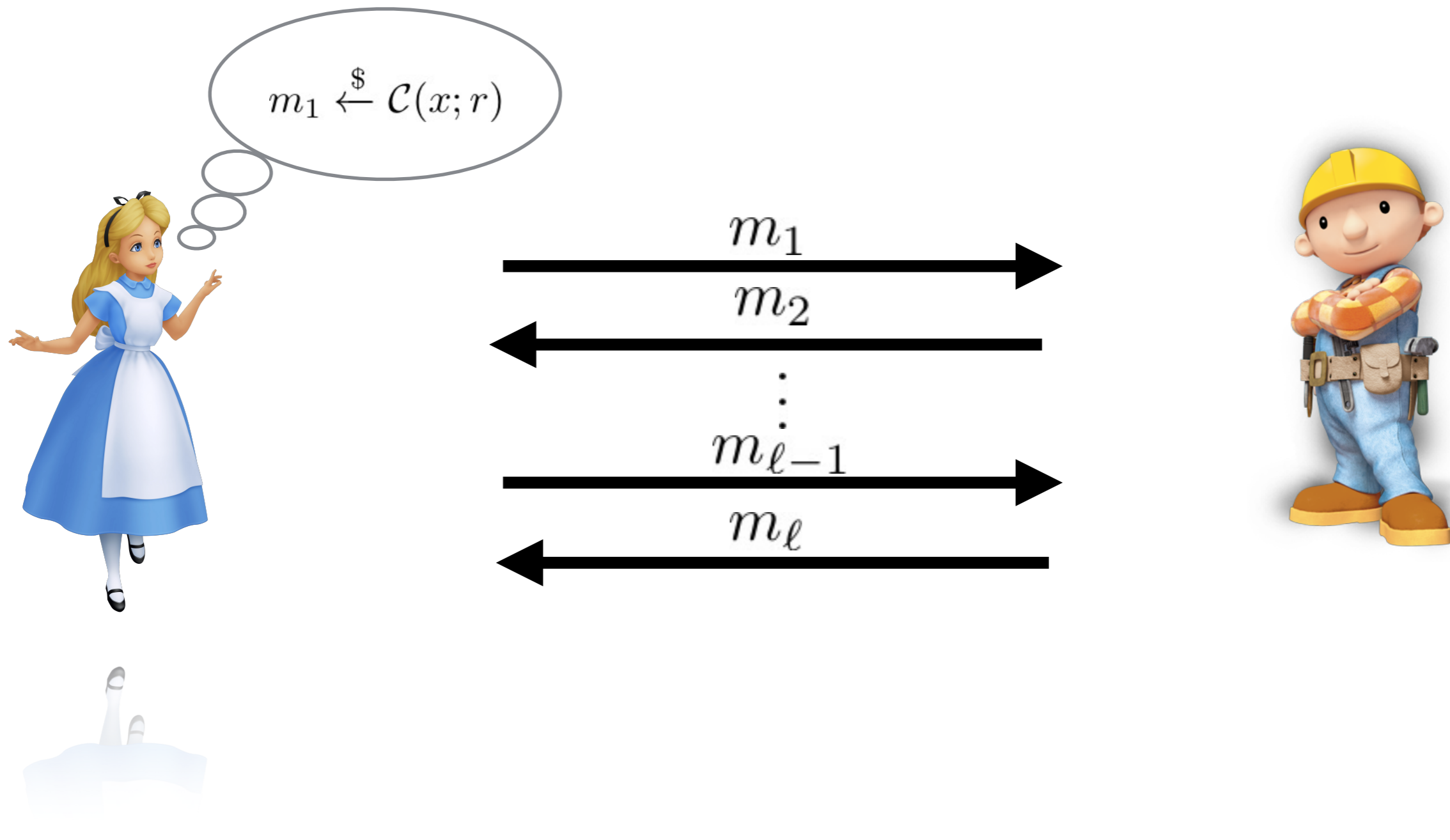
# Classical Crypto



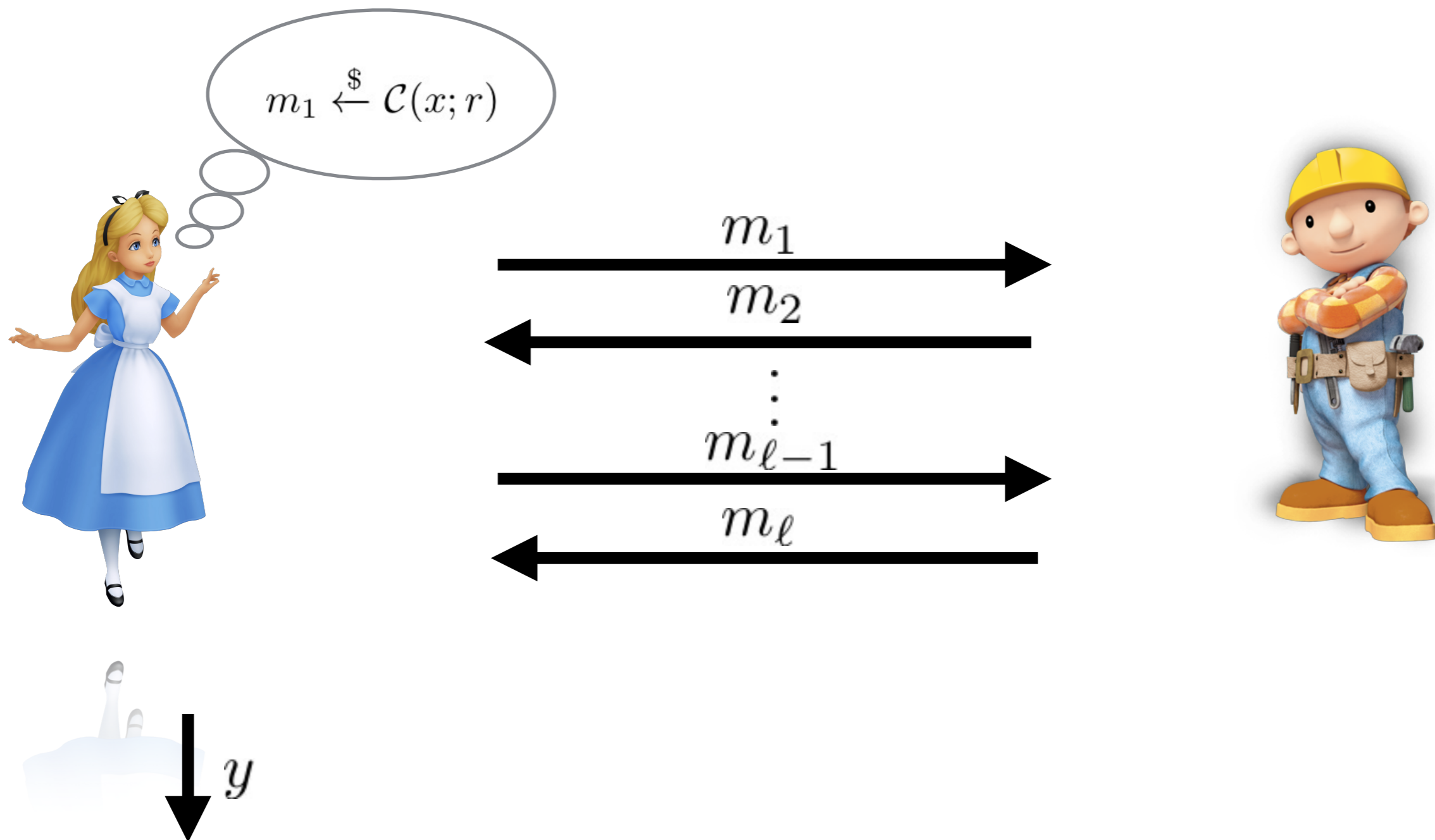
# Classical Crypto



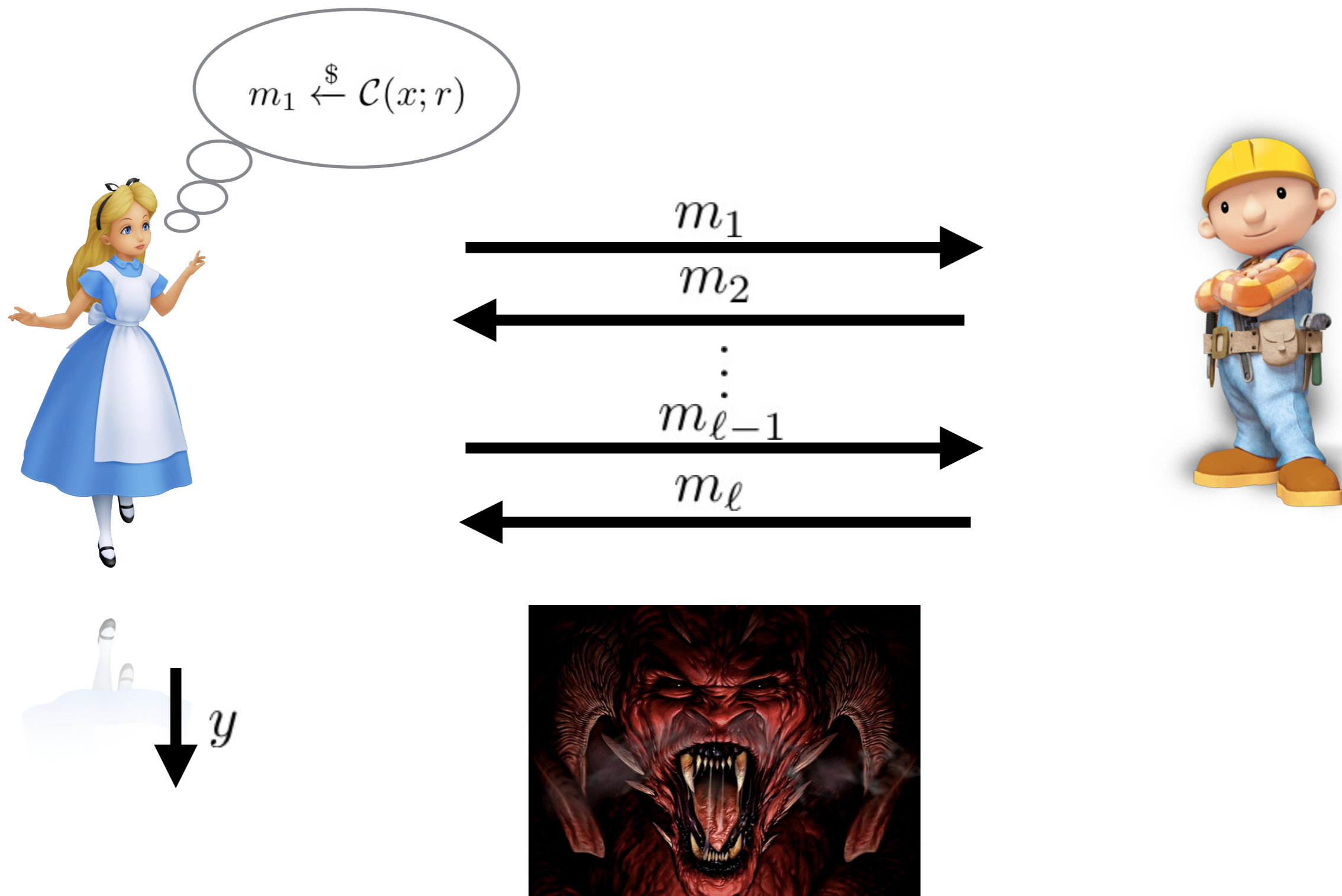
# Classical Crypto



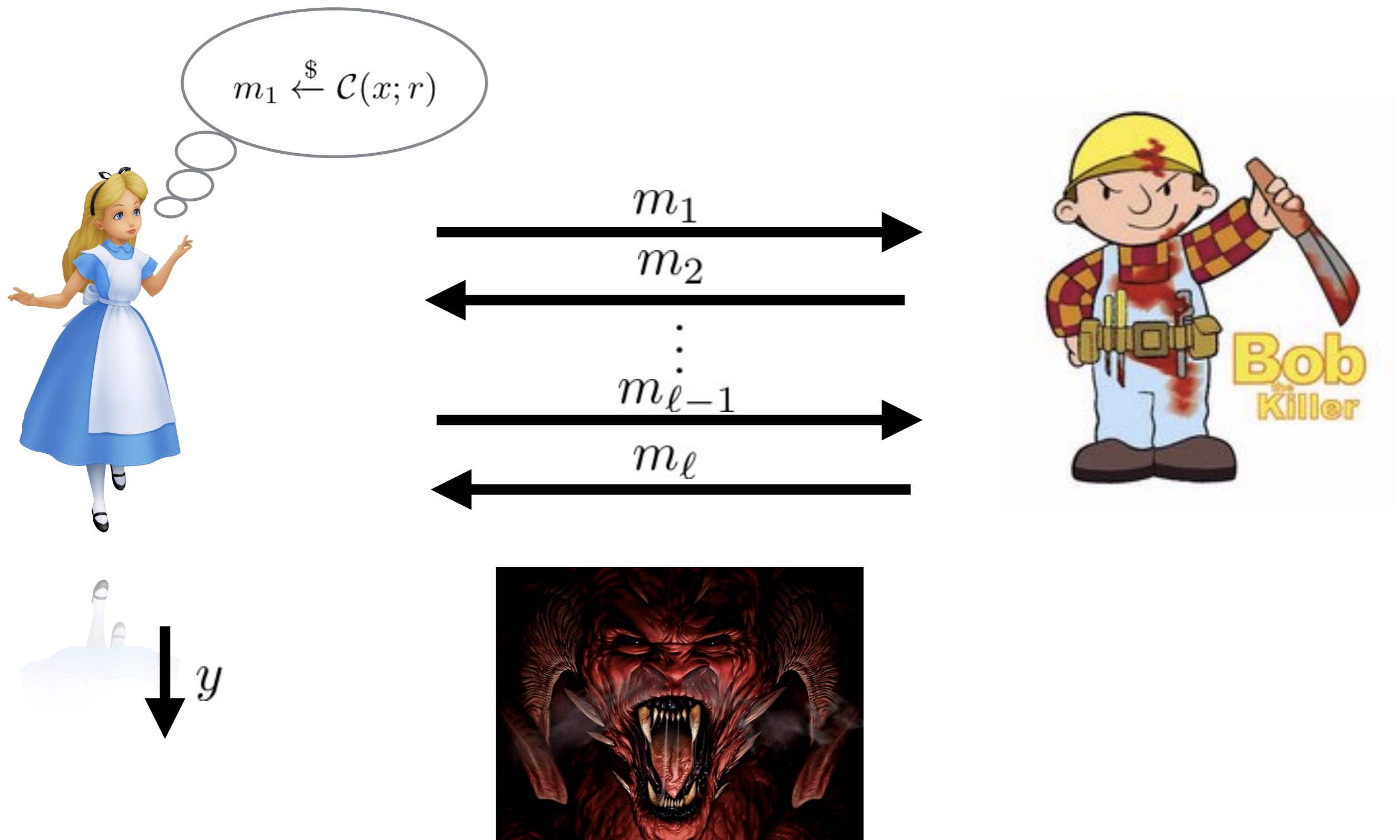
# Classical Crypto



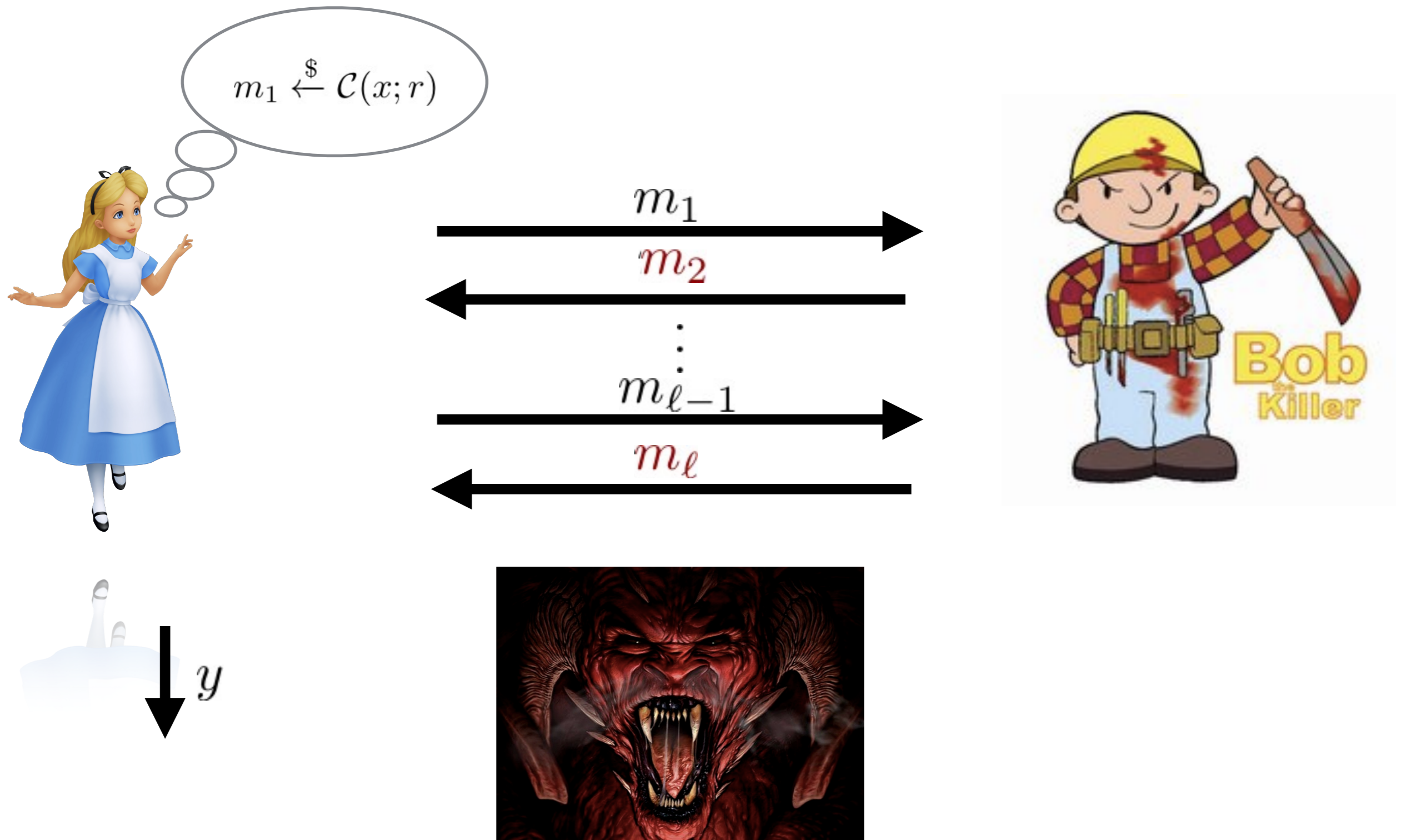
# Classical Crypto



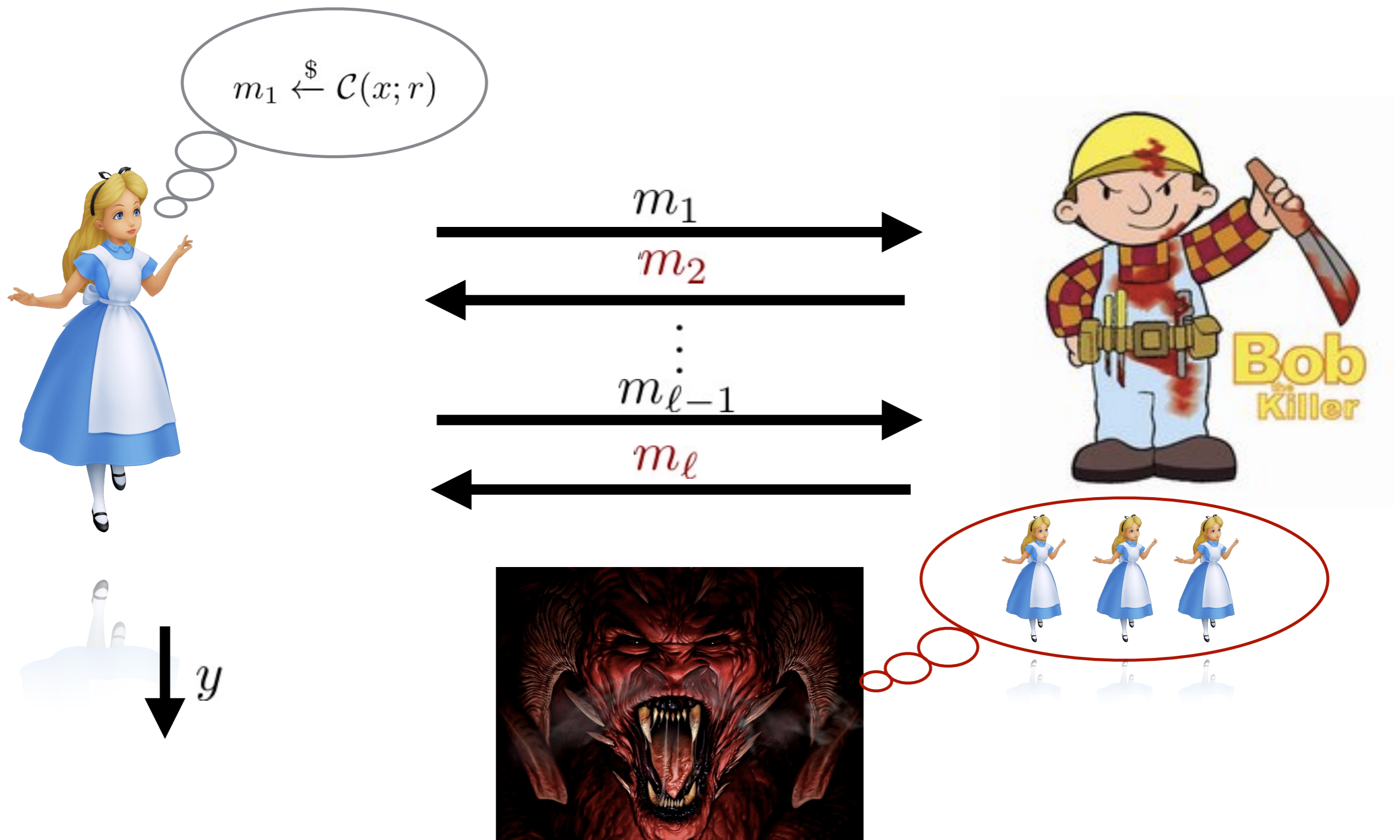
# Classical Crypto



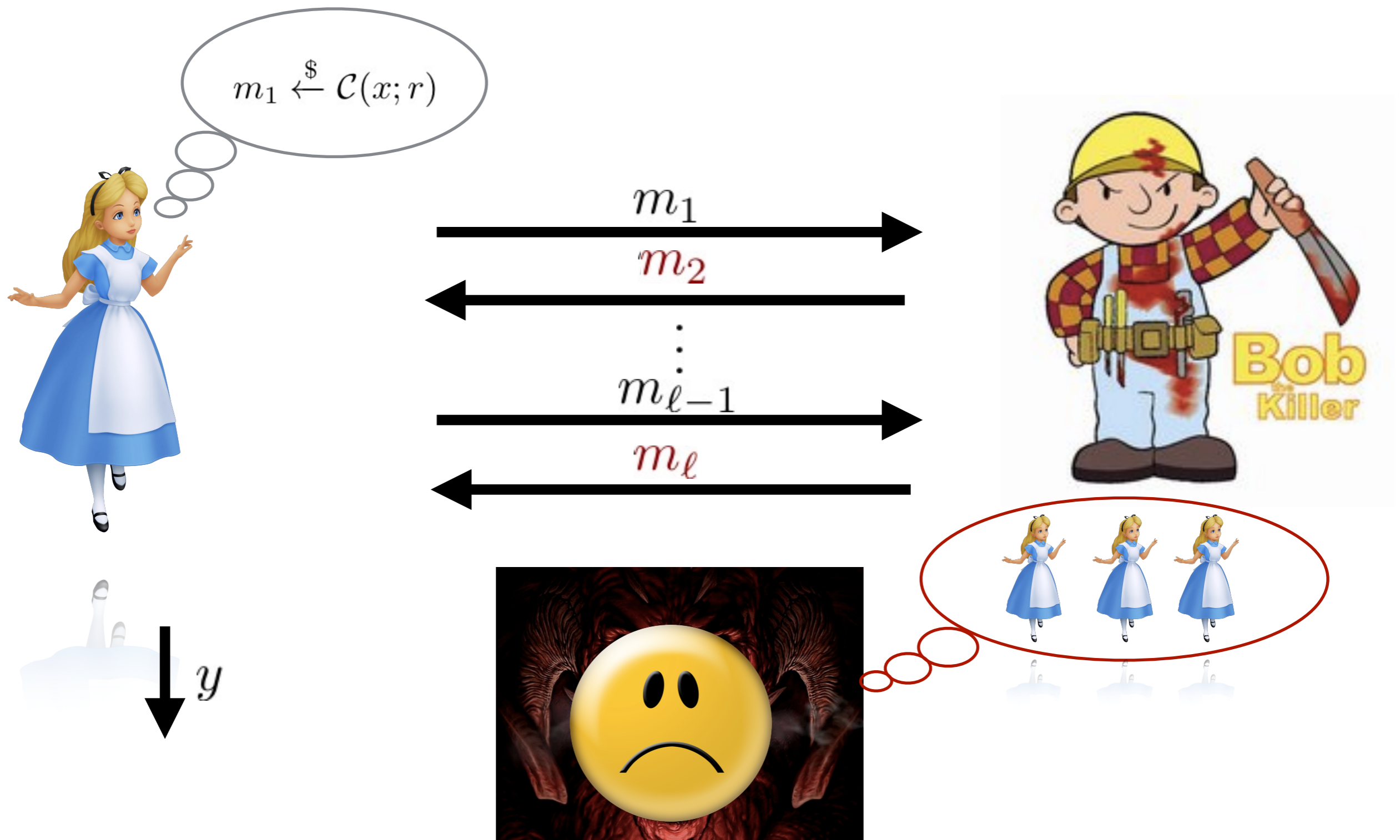
# Classical Crypto



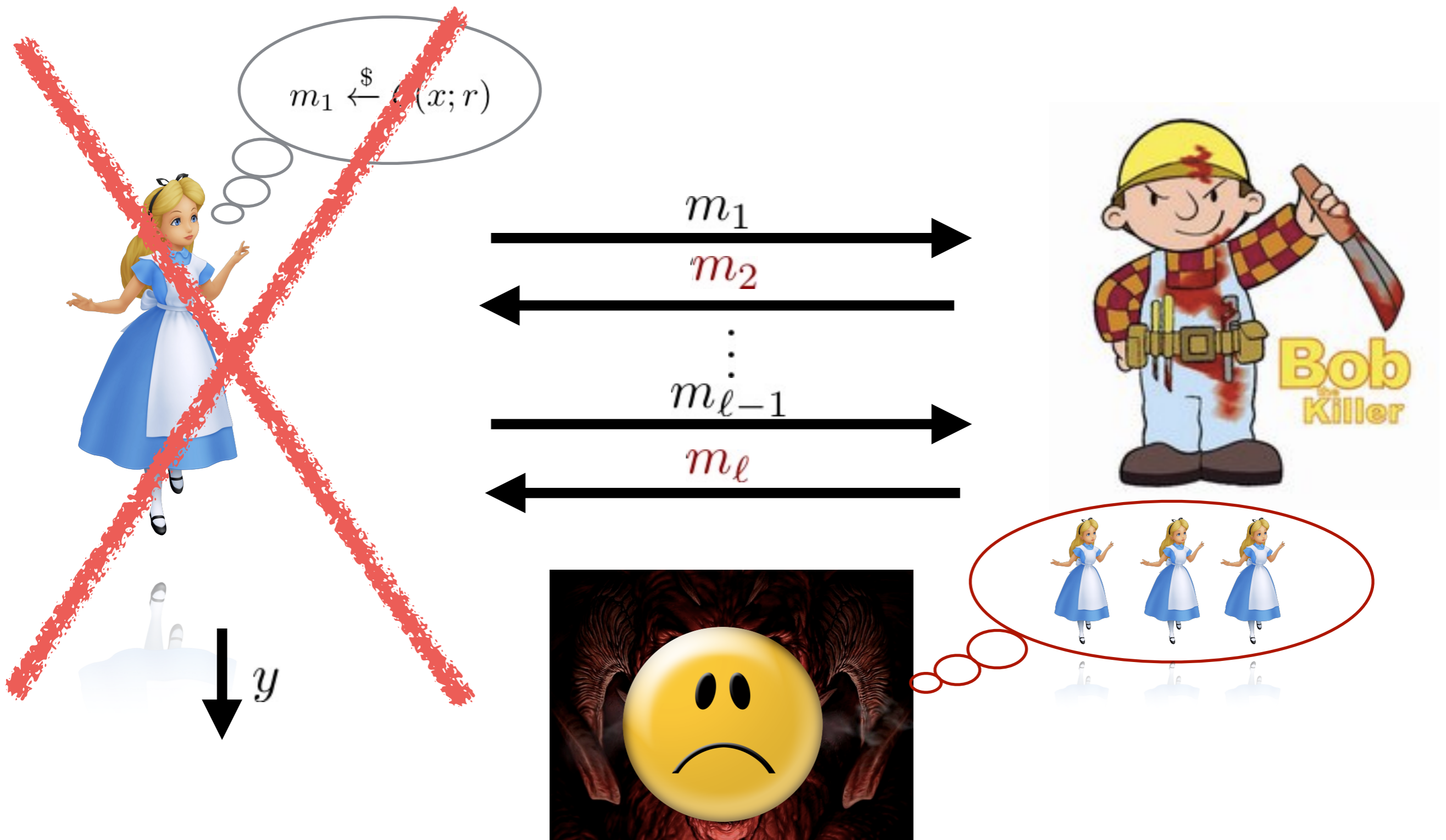
# Classical Crypto



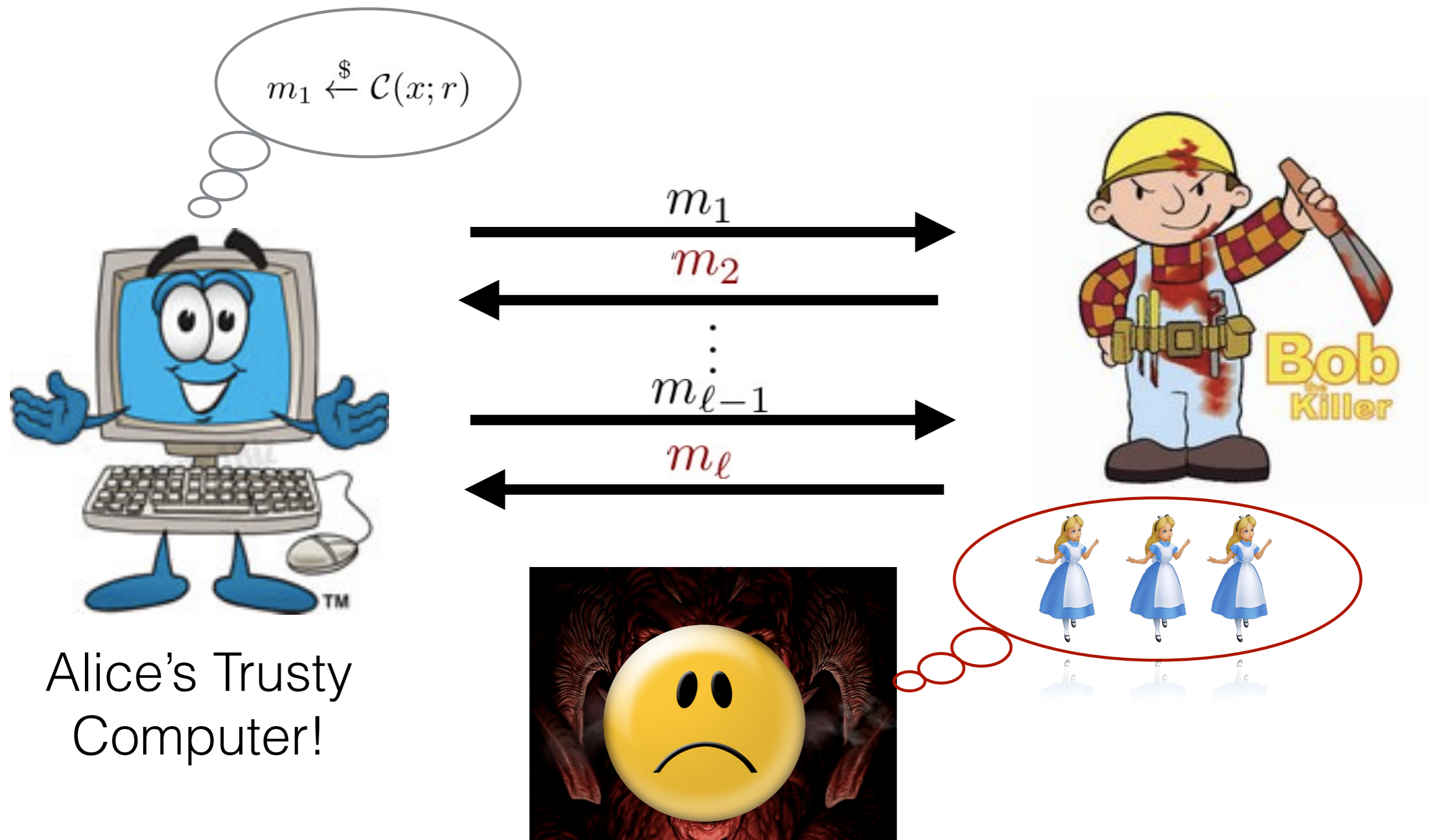
# Classical Crypto



# Classical Crypto



# Classical Crypto



Alice's Trusty  
Computer!

# Should Alice Trust Her Computer?

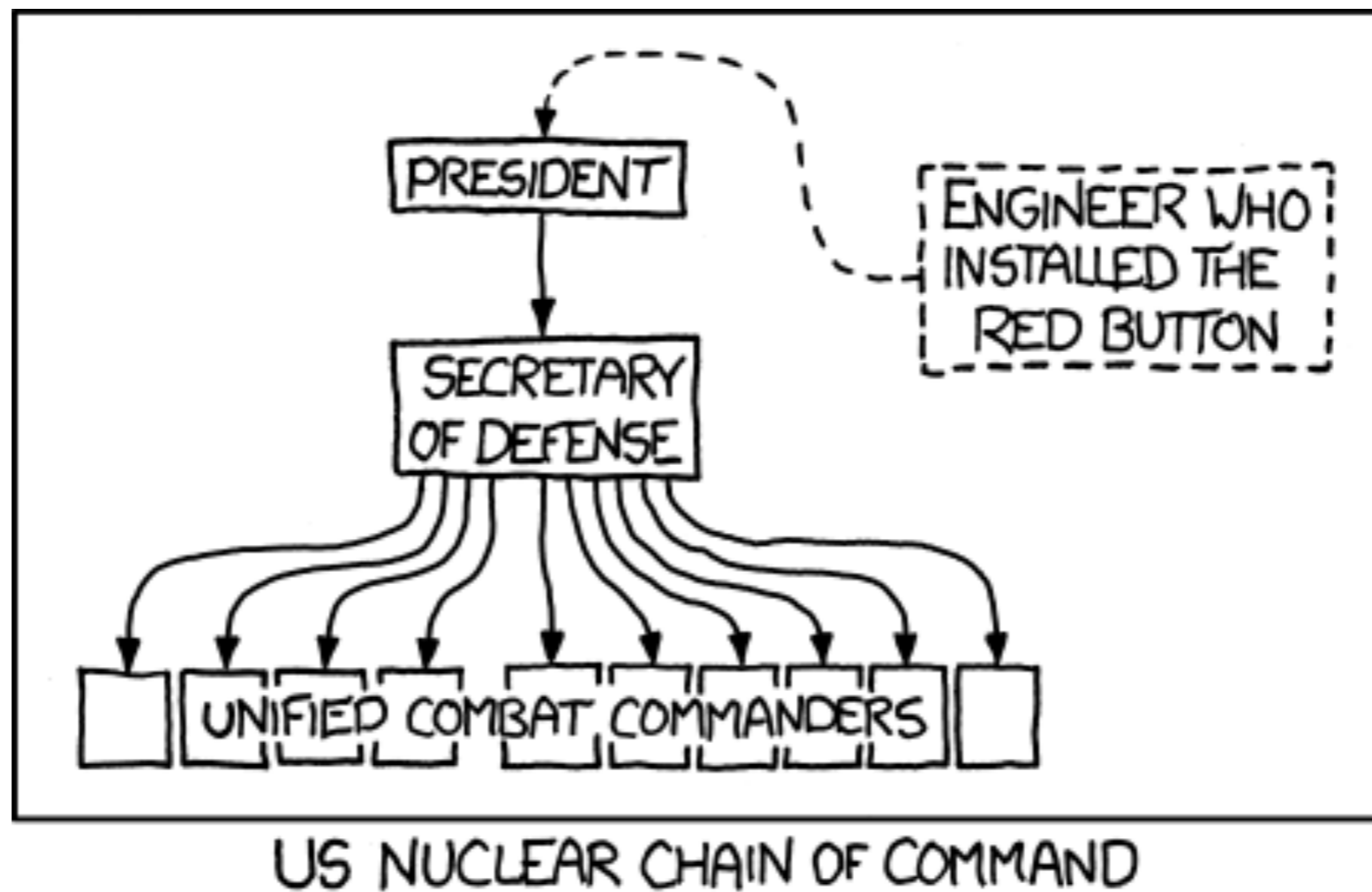
# Should Alice Trust Her Computer?



# Should Alice Trust Her Computer?



# Should Alice Trust Her Computer?



([xkcd.com](http://xkcd.com))

# Widespread Deliberate Corruption of Hardware and Software

# Widespread Deliberate Corruption of Hardware and Software



# Widespread Deliberate Corruption of Hardware and Software

“The SIGINT Enabling Project [\$250M/year program] actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products’ designs. These design changes make the systems in question exploitable ... with foreknowledge of the modification. To the consumer and other adversaries, however, the systems’ security remain intact.”

Excerpt from the N.S.A.’s 2013 budget request

The New York *Times*, September 5, 2013

# Last Year, We Agreed Not to Accept This

The membership of the IACR repudiates mass surveillance and the undermining of cryptographic solutions and standards. Population-wide surveillance threatens democracy and human dignity. We call for expediting research and deployment of effective techniques to protect personal privacy against governmental and corporate overreach.

IACR Copenhagen Resolution  
Eurocrypt 2014, Copenhagen

# Widespread, (Apparently) Accidental Bugs in Cryptographic Software

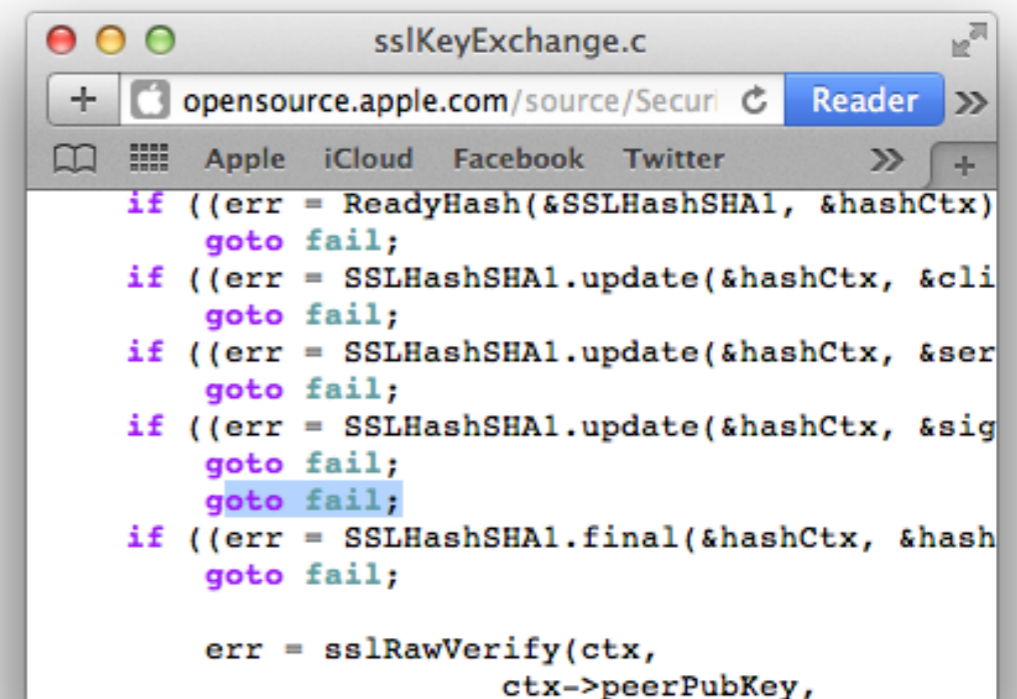
# Widespread, (Apparently) Accidental Bugs in Cryptographic Software



# Widespread, (Apparently) Accidental Bugs in Cryptographic Software



# Widespread, (Apparently) Accidental Bugs in Cryptographic Software

A screenshot of a code editor window titled "sslKeyExchange.c". The code is in C and shows several "if" statements checking for errors. The line "goto fail;" is highlighted in blue in the second "if" statement. The code is as follows:

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &ser
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hash
    goto fail;

err = sslRawVerify(ctx,
                  ctx->peerPubKey,
```

# Widespread, (Apparently) Accidental Bugs in Cryptographic Software



## Debian Security Advisory

DSA-1571-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &ser
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hash
    goto fail;

err = sslRawVerify(ctx,
                  ctx->peerPubKey,
```



# Widespread, (Apparently) Accidental Bugs in Cryptographic Software



## Debian Security Advisory

DSA-1571-1 openssl -- predictable random number generator

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx))
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &cli
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &ser
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &sig
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hash
    goto fail;
sslRawVerify(ctx,
    ctx->peerPubKey,
```

## Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger<sup>†\*</sup>

Zakir Durumeric<sup>‡\*</sup>

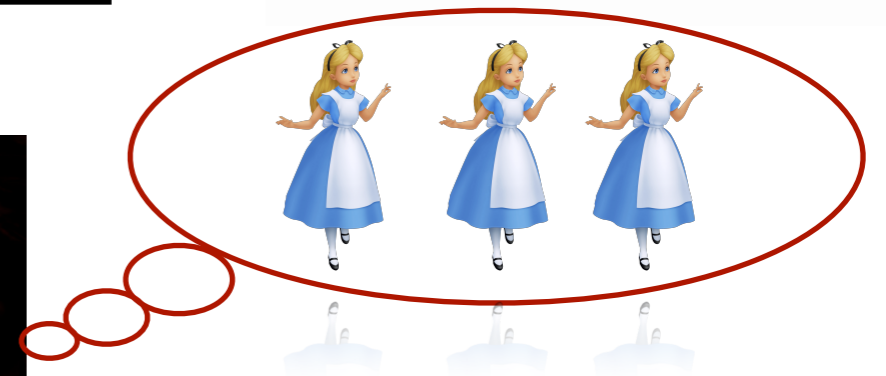
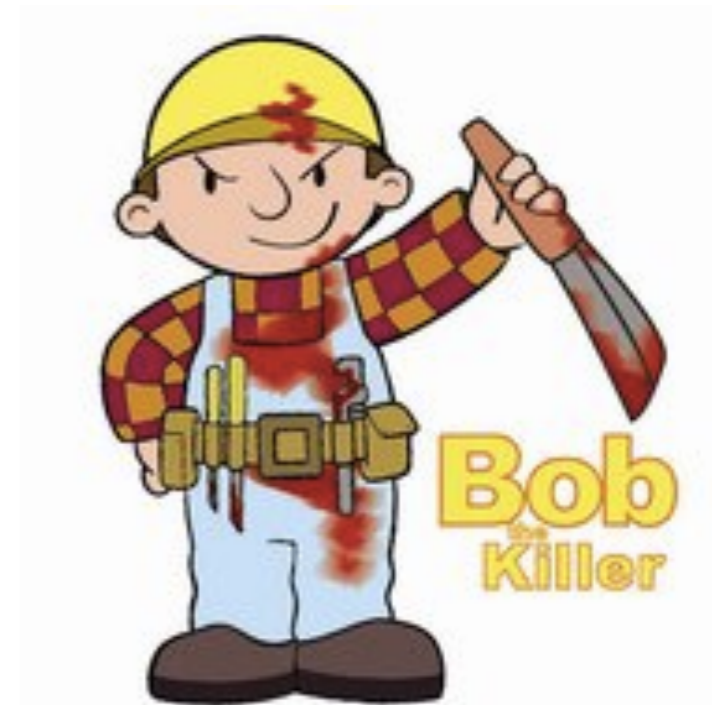
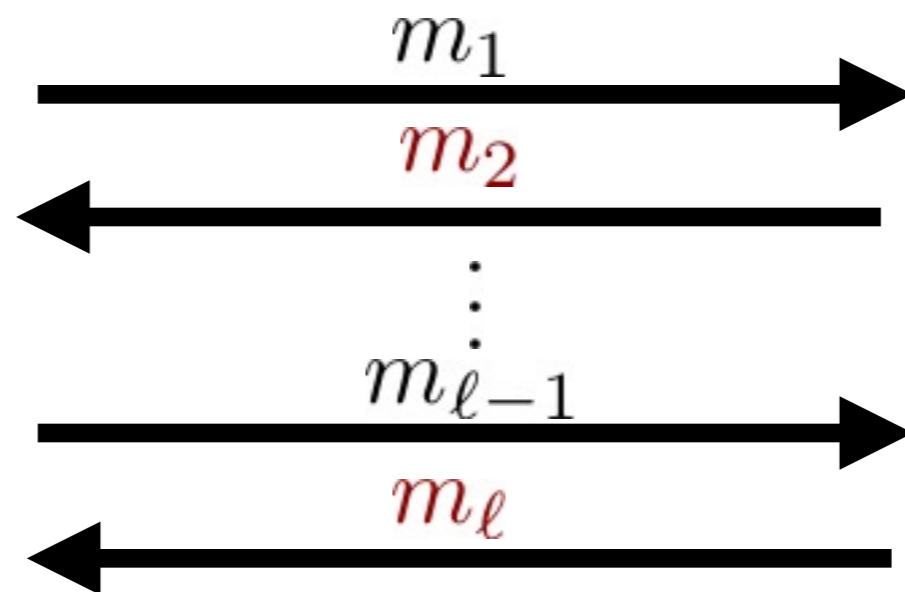
Eric Wustrow<sup>‡</sup>

J. Alex Halderman<sup>‡</sup>

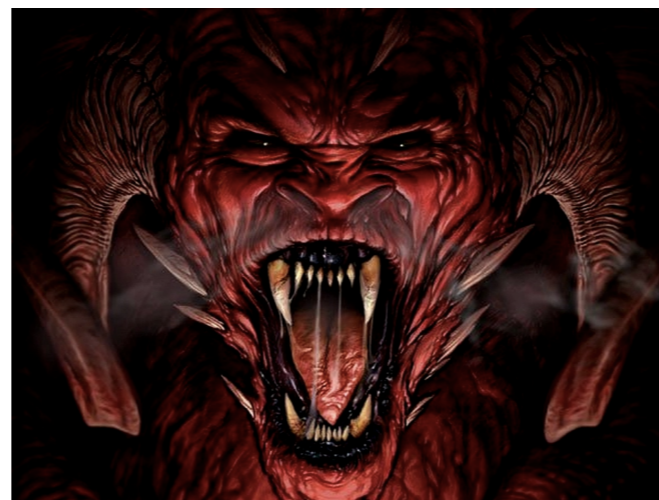
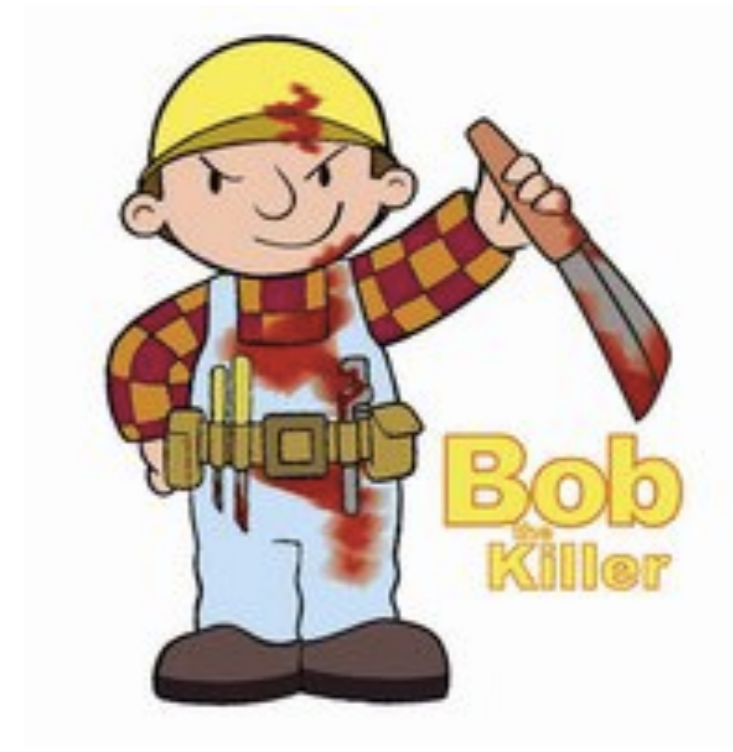
<sup>†</sup> *University of California, San Diego*  
nadiiah@cs.ucsd.edu

<sup>‡</sup> *The University of Michigan*  
{zakir, ewust, jhalderm}@umich.edu

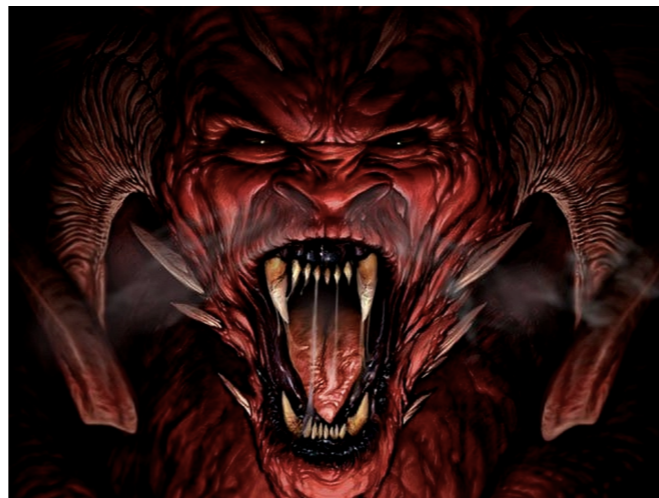
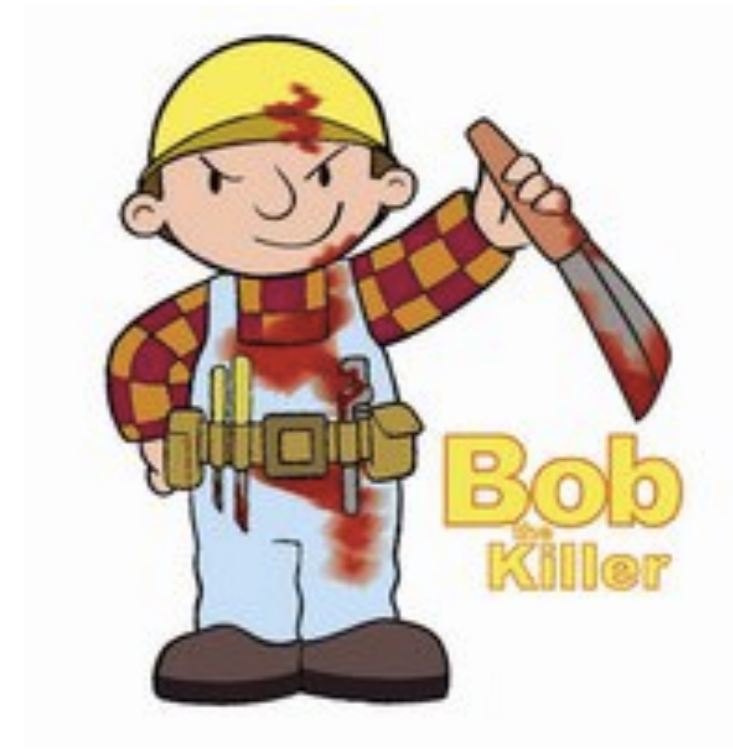
# Crypto in the Real World?



# Crypto in the Real World?

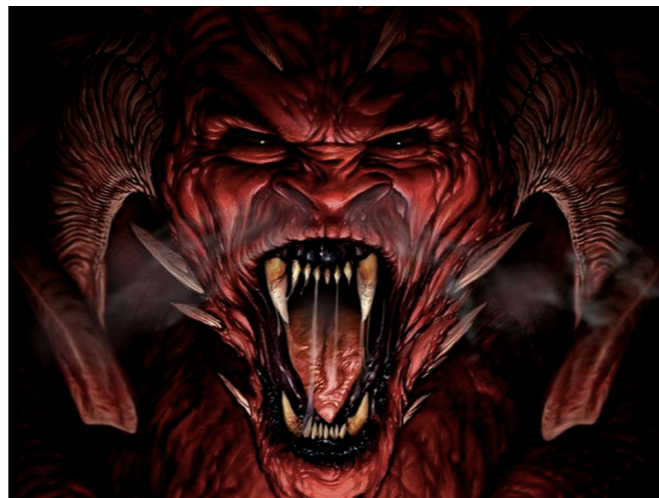


# Crypto in the Real World?



# Crypto in the Real World?

iHEARTdogs&cat\$



# Crypto in the Real World?



iHEARTdogs&cat\$

4117-8289-1856

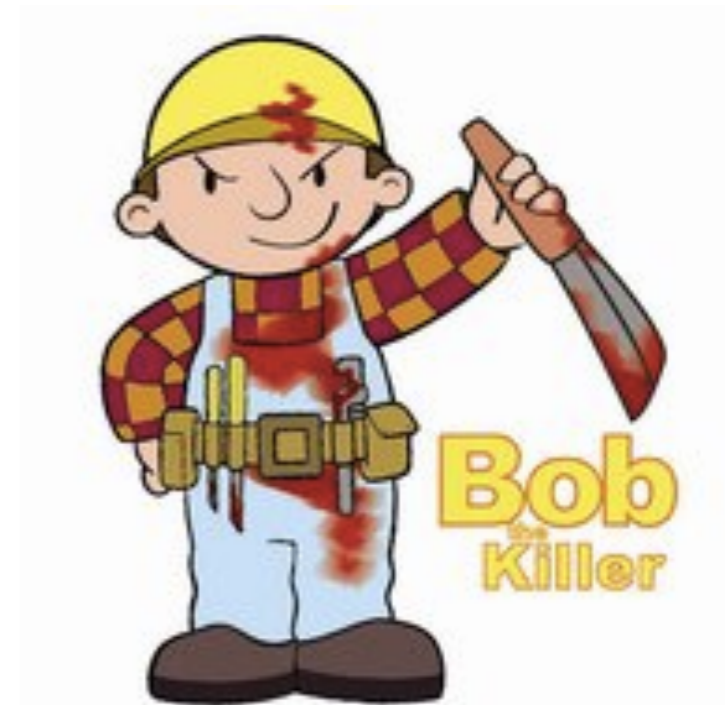


# Crypto in the Real World?



iHEARTdogs&cat\$

4117-8289-1856



# Crypto in the Real World?

iHEARTdogs&cat\$



Steganography, randomness subversion, backdoored PRGs, etc.



Can we possibly do crypto  
on a compromised machine?

# Prior Work

# Prior Work

- Subliminal channels

# Prior Work

- Subliminal channels
  - Simmons 1984, ...

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE
    - Limited forms of corruption

# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE
    - Limited forms of corruption
- Backdoored PRGs

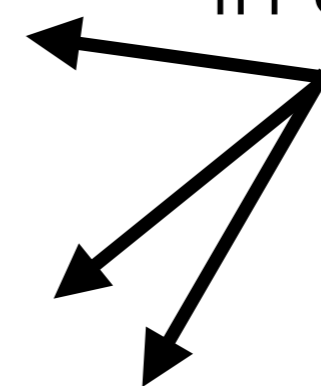
# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE
    - Limited forms of corruption
- Backdoored PRGs
  - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015 (talk Monday)

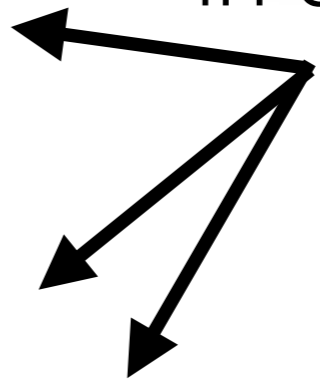
# Prior Work

- Subliminal channels
  - Simmons 1984, ...
- Divertible protocols
  - Blaze, Bleumer, Strauss 1998
    - Limited security against limited forms of corruption
    - Only synchronous protocols
- Kleptography and cryptovirology
  - Young and Yung 1996
- Algorithm Substitution Attacks
  - Bellare, Paterson, Rogaway 2014
    - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE
    - Limited forms of corruption
- Backdoored PRGs
  - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015 (talk Monday)

Impossibility results  
in strongest models



# Prior Work

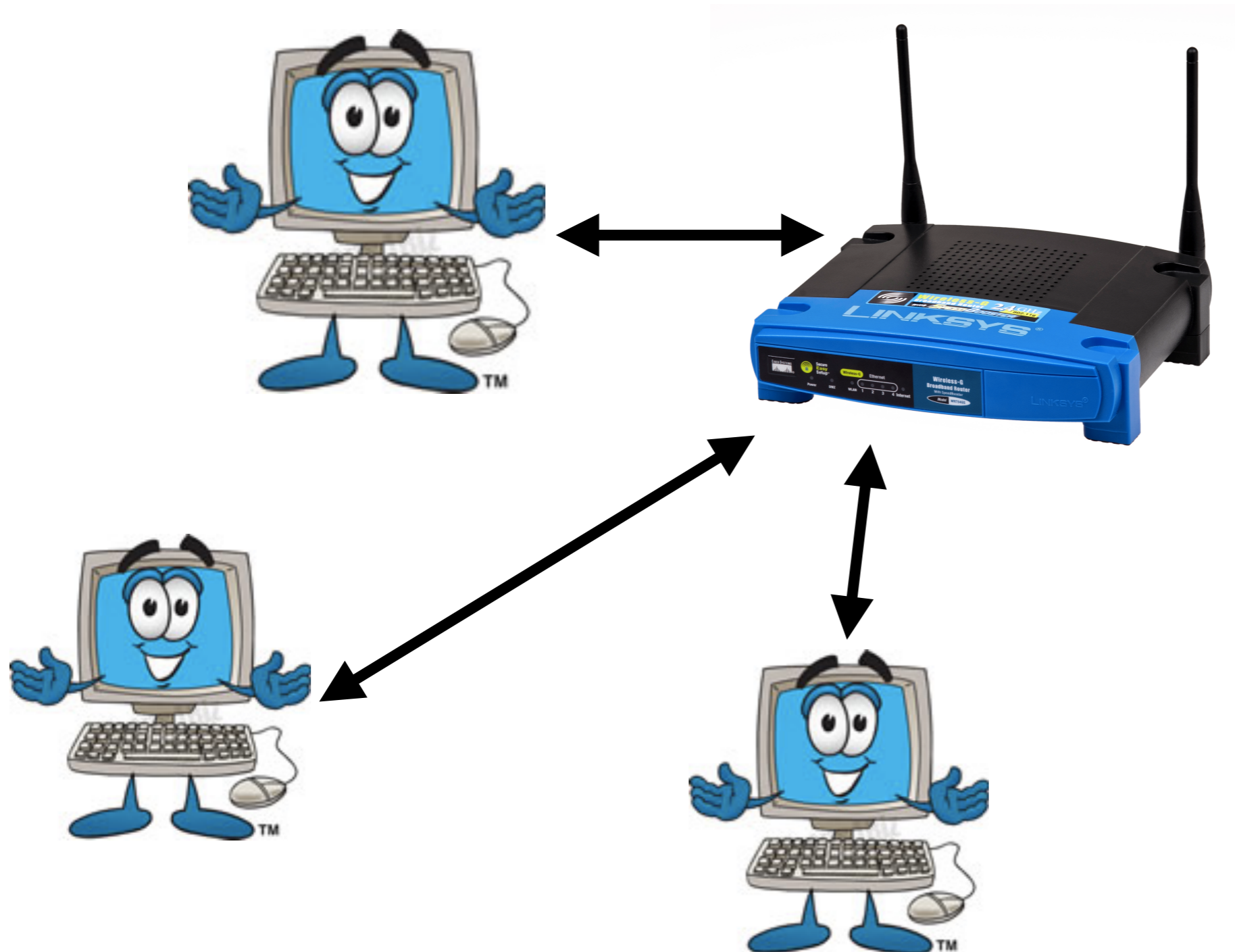
- Subliminal channels
    - Simmons 1984, ...
  - Divertible protocols
    - Blaze, Bleumer, Strauss 1998
      - Limited security against limited forms of corruption
      - Only synchronous protocols
  - Kleptography and cryptovirology
    -
  - Alg We generalize these models and show a way around the impossibility results!
  - - Symmetric encryption
    - Limited forms of corruption
  - Bellare and Hoang 2015 (previous talk)
    - Deterministic PKE
    - Limited forms of corruption
  - Backdoored PRGs
    - Dodis, Ganesh, Golovnev, Juels, and Ristenpart 2015 (talk Monday)
- 

# Reverse Firewalls!

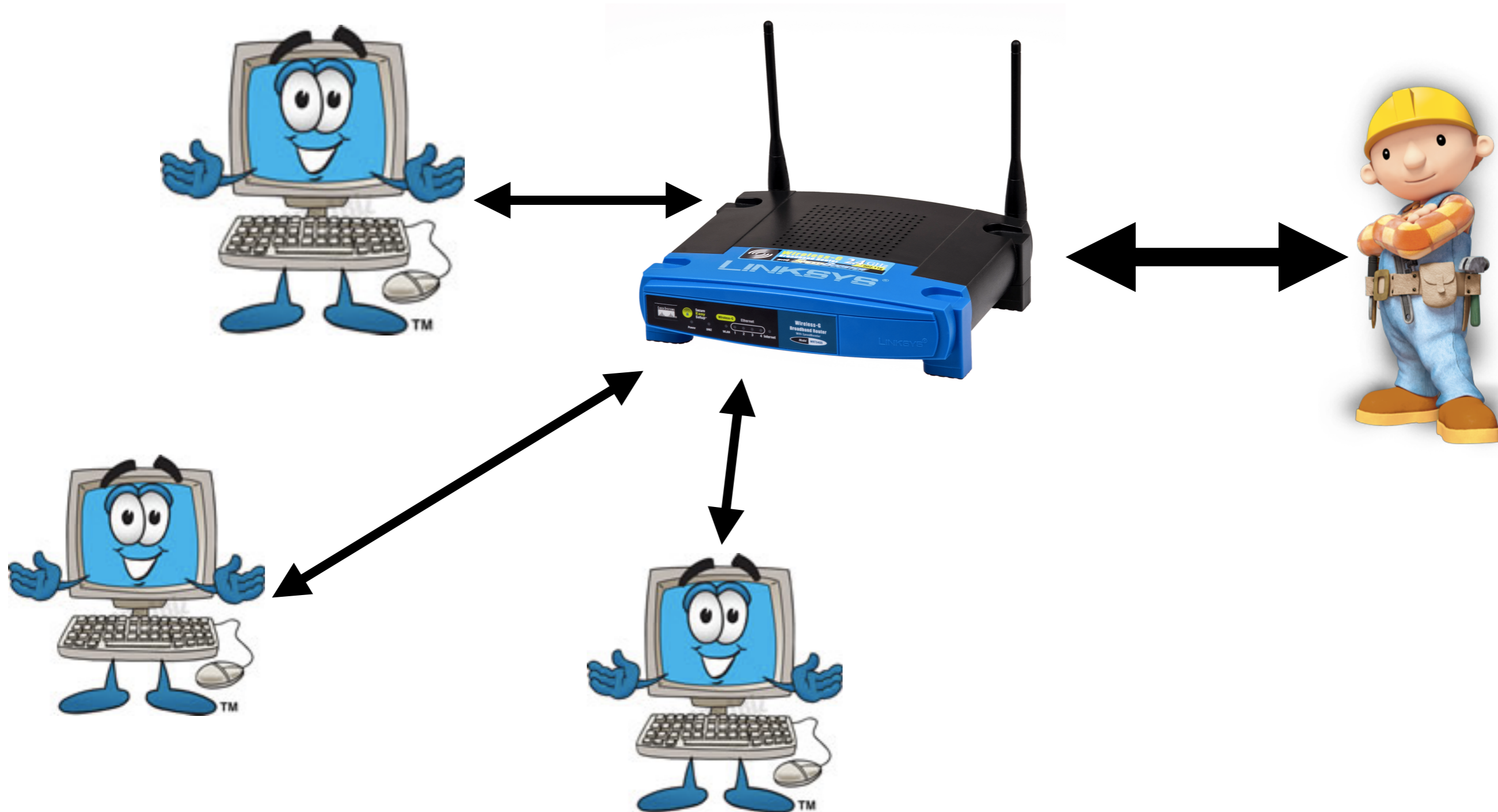
# Reverse Firewalls!



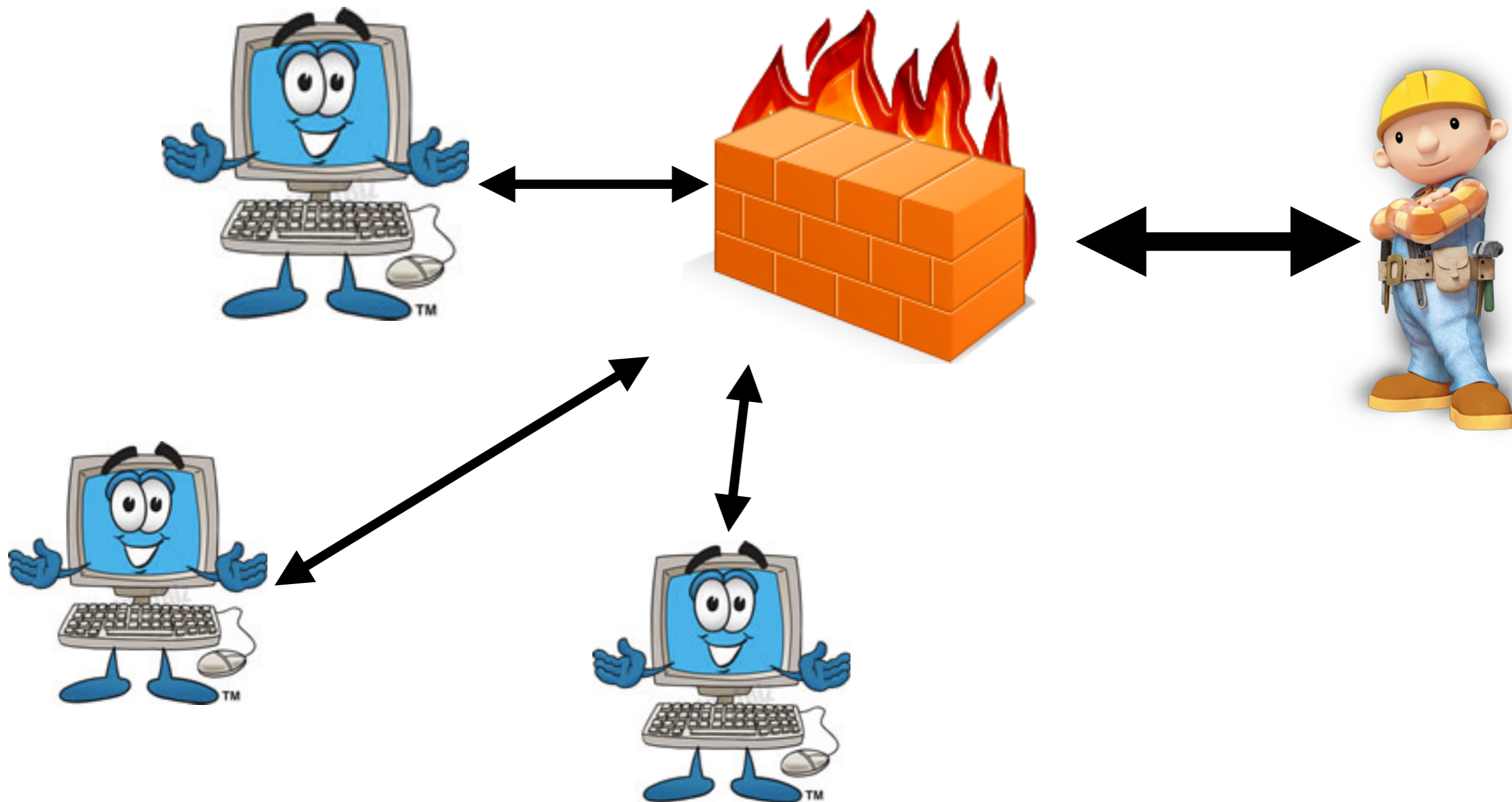
# Reverse Firewalls!



# Reverse Firewalls!



# Reverse Firewalls!



# Reverse Firewalls!

# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.

# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



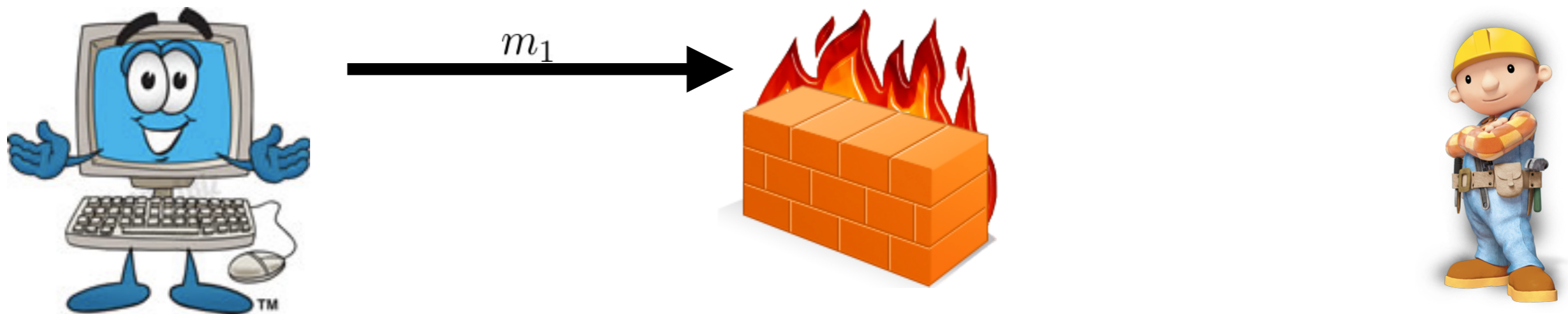
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



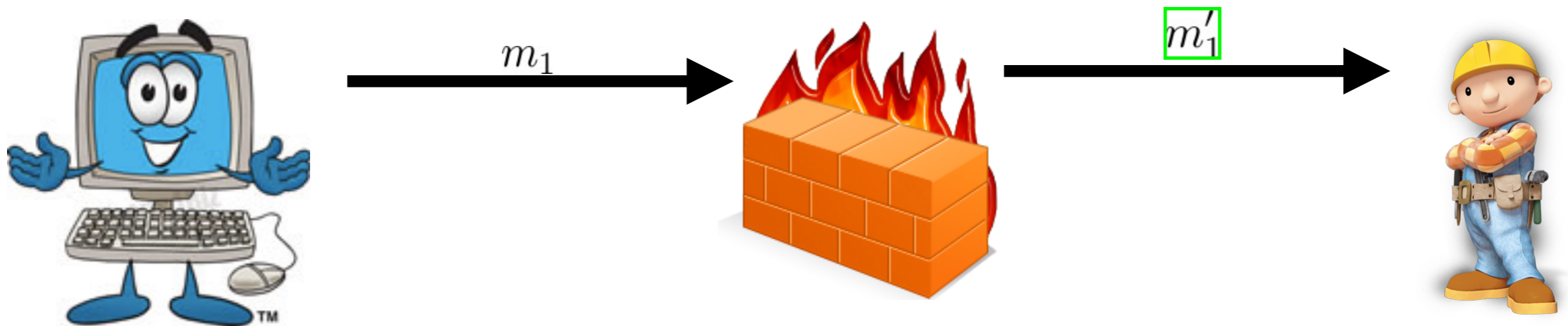
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



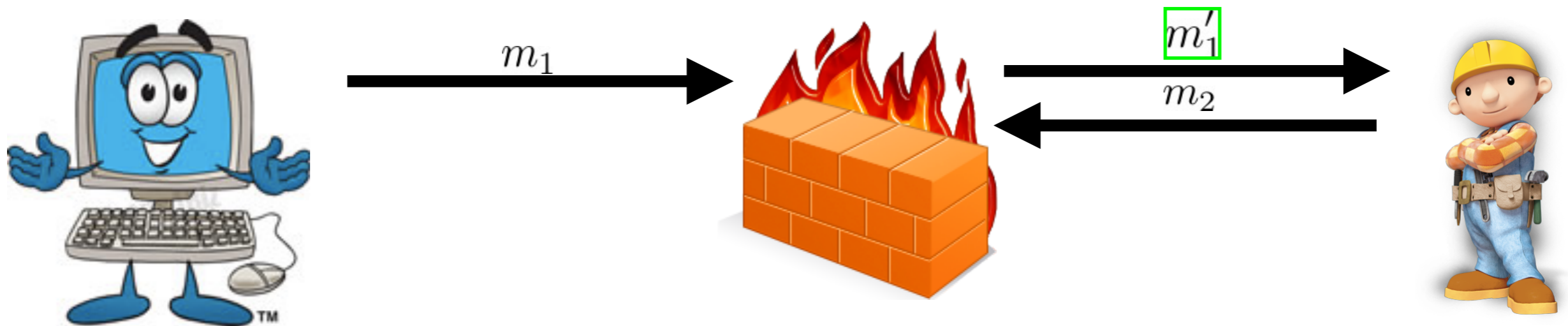
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



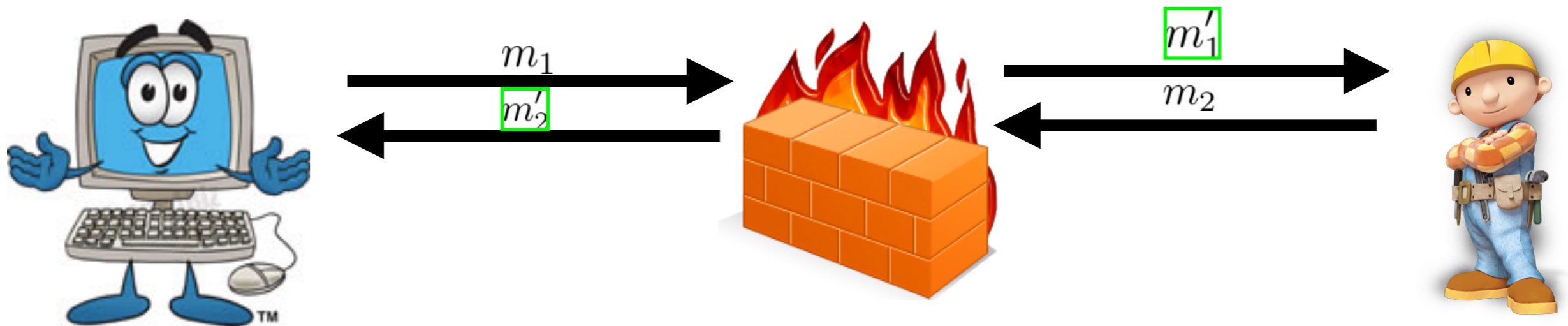
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



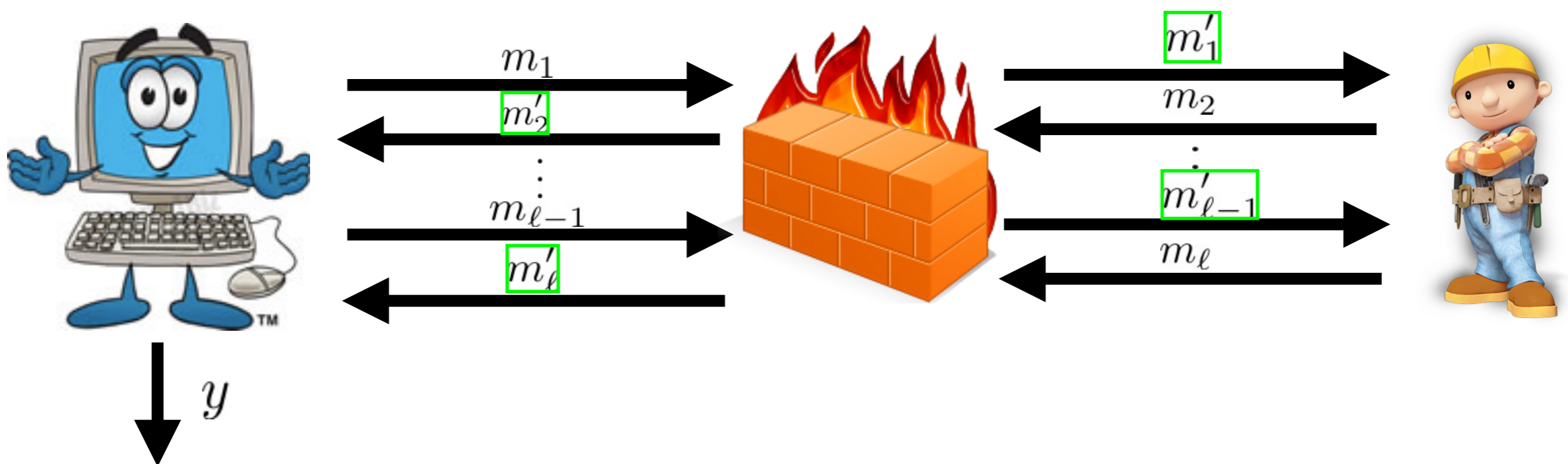
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



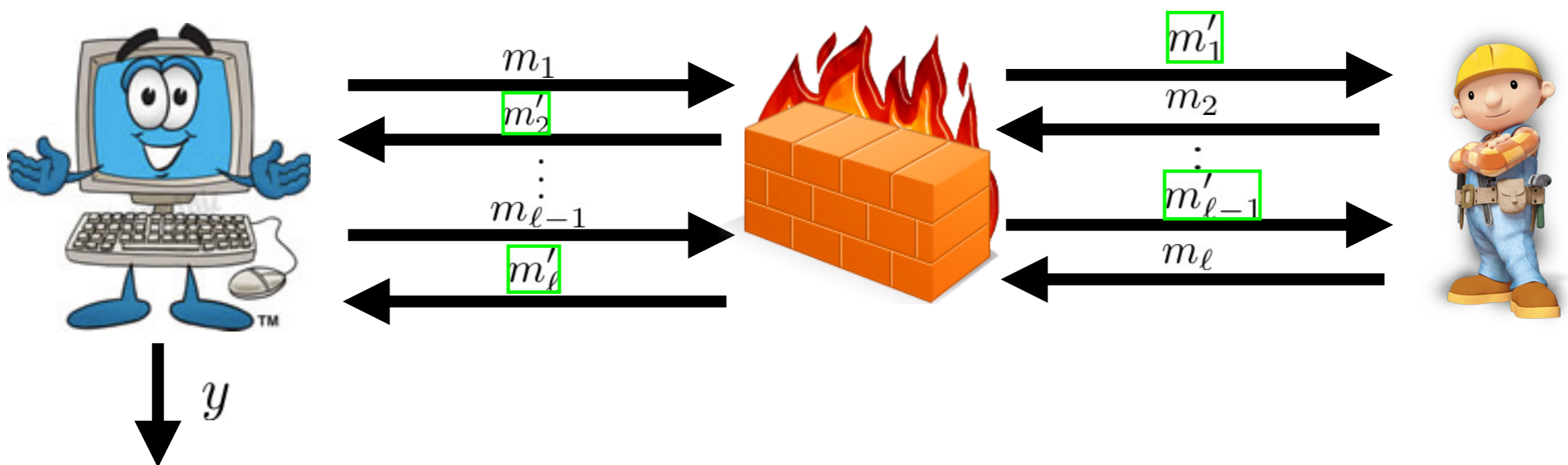
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.



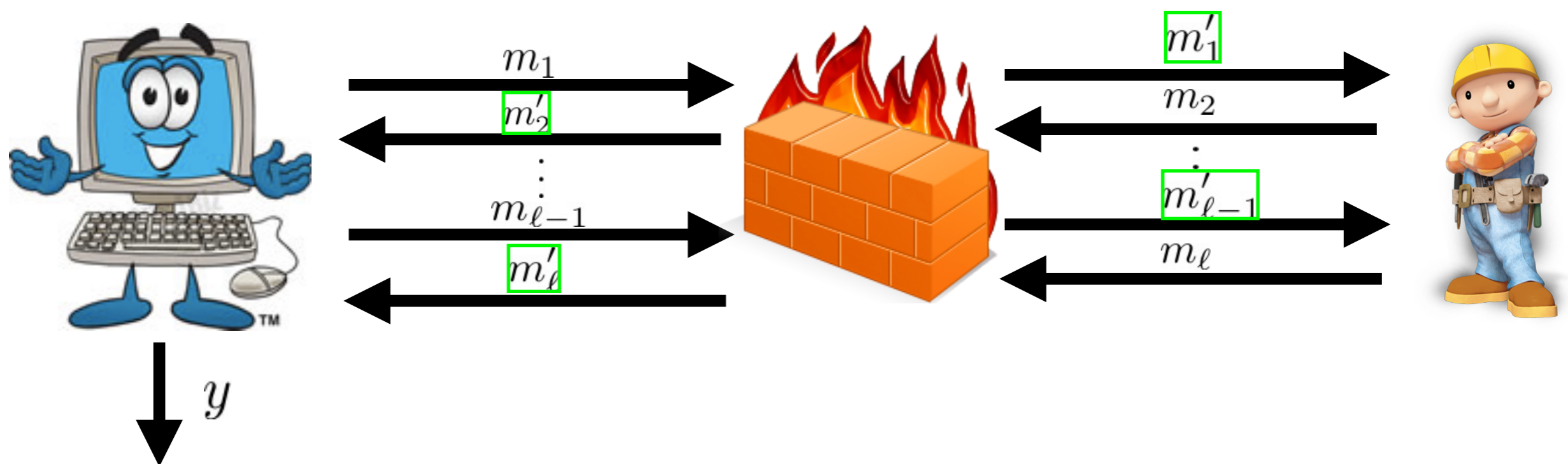
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.



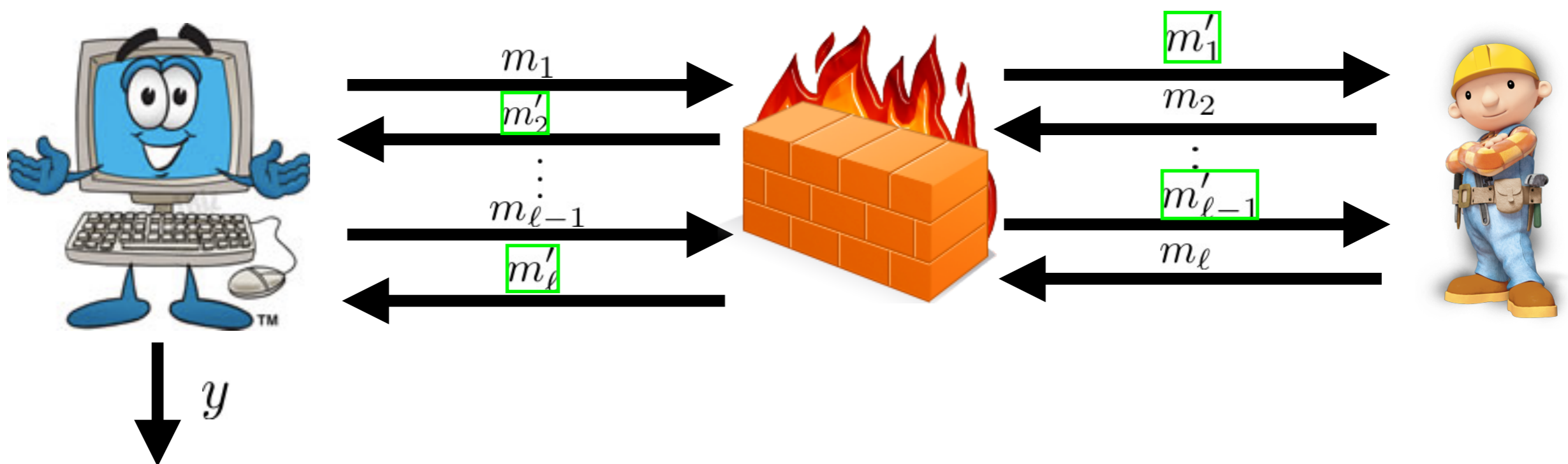
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
  - Certainly doesn't break functionality.



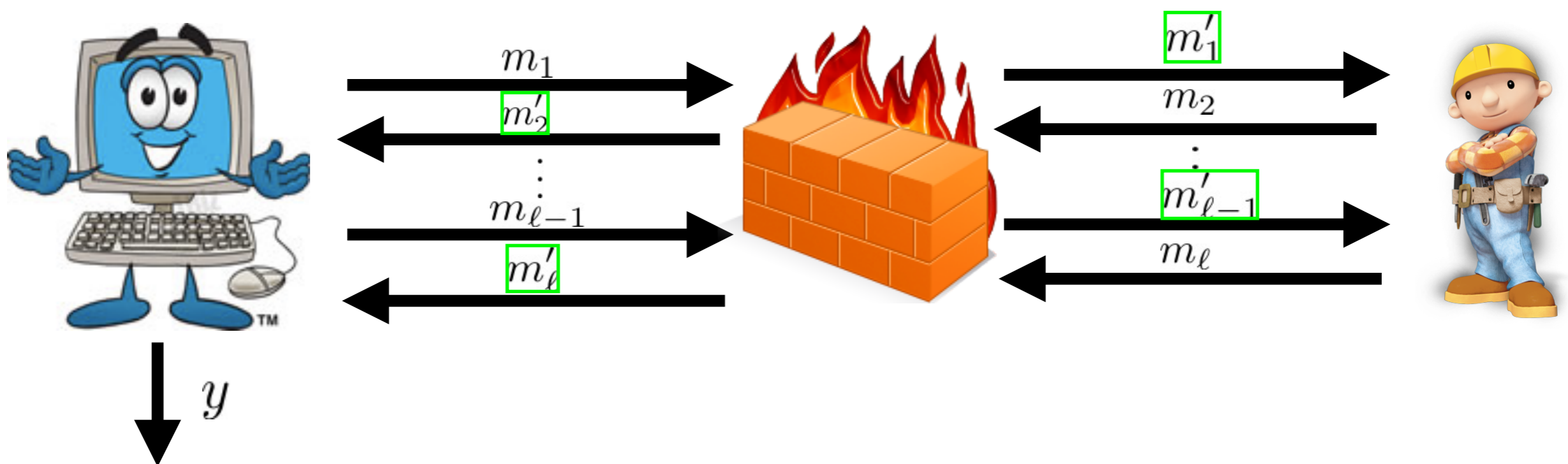
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
  - Certainly doesn't break functionality.
- Shares no secrets with Alice.



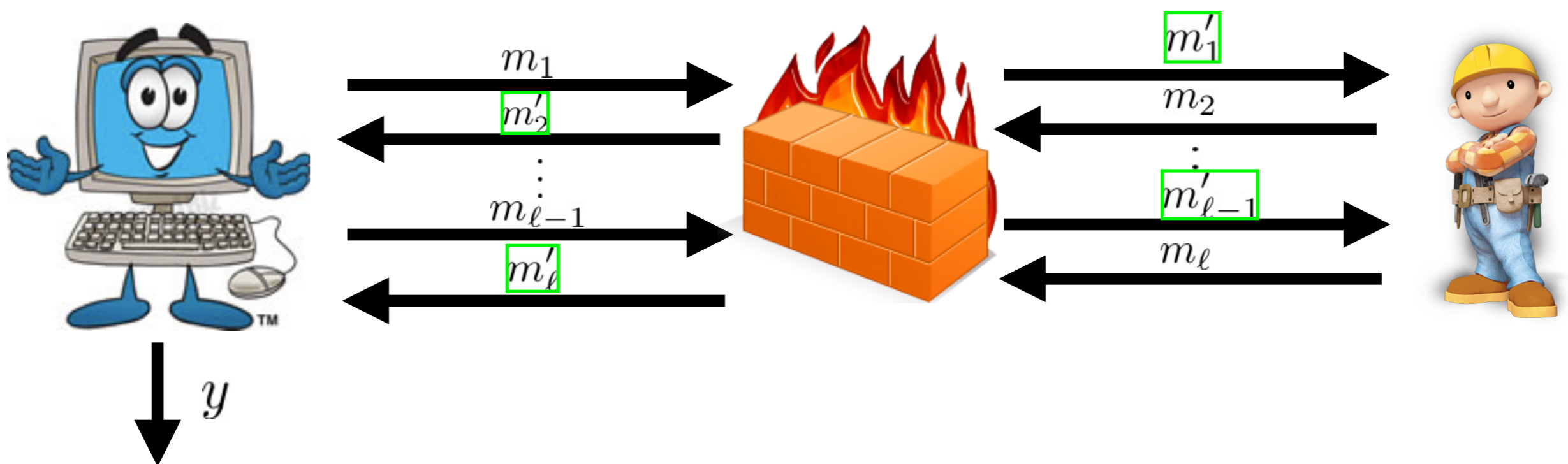
# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
  - Certainly doesn't break functionality.
- Shares no secrets with Alice.
  - **We don't trust the firewall.**



# Reverse Firewalls!

- Modifies the messages that Alice sends and receives.
- Transparent to legitimate traffic.
  - Certainly doesn't break functionality.
- Shares no secrets with Alice.
  - **We don't trust the firewall.**
- “Improves security!”



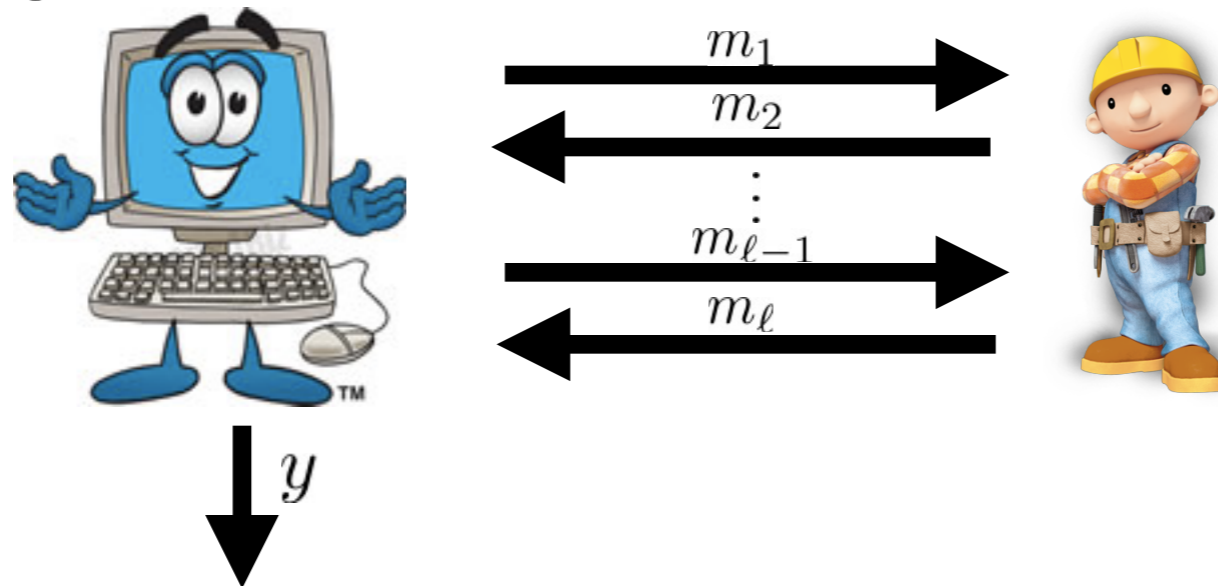
# Reverse Firewall Functionality

# Reverse Firewall Functionality

Underlying classical protocol has some functionality.

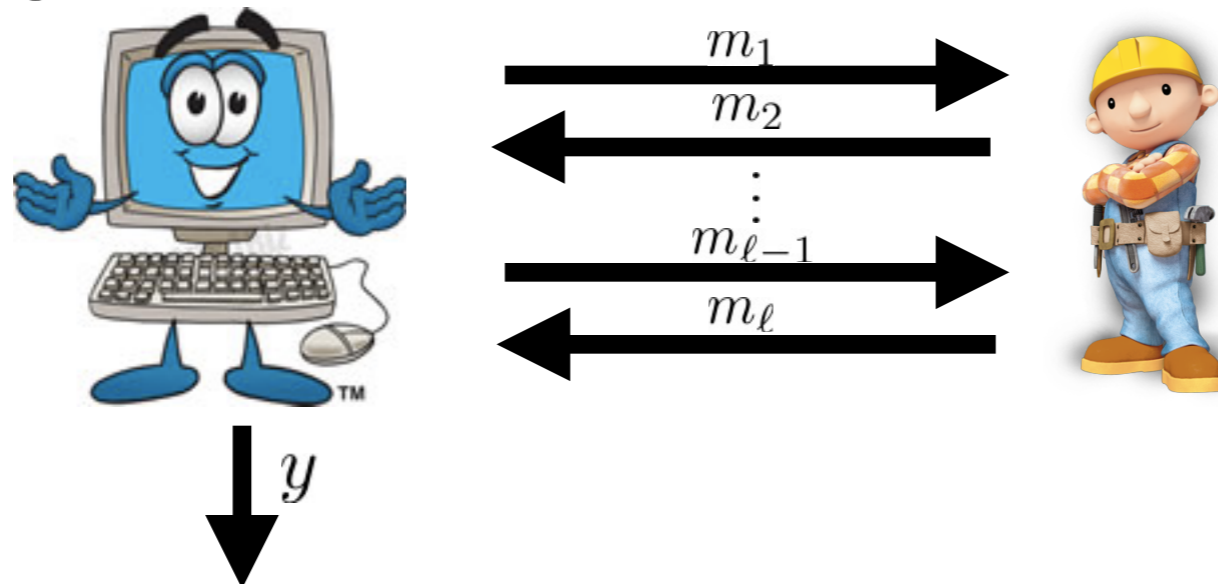
# Reverse Firewall Functionality

Underlying classical protocol has some functionality.



# Reverse Firewall Functionality

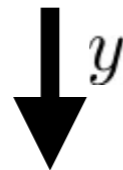
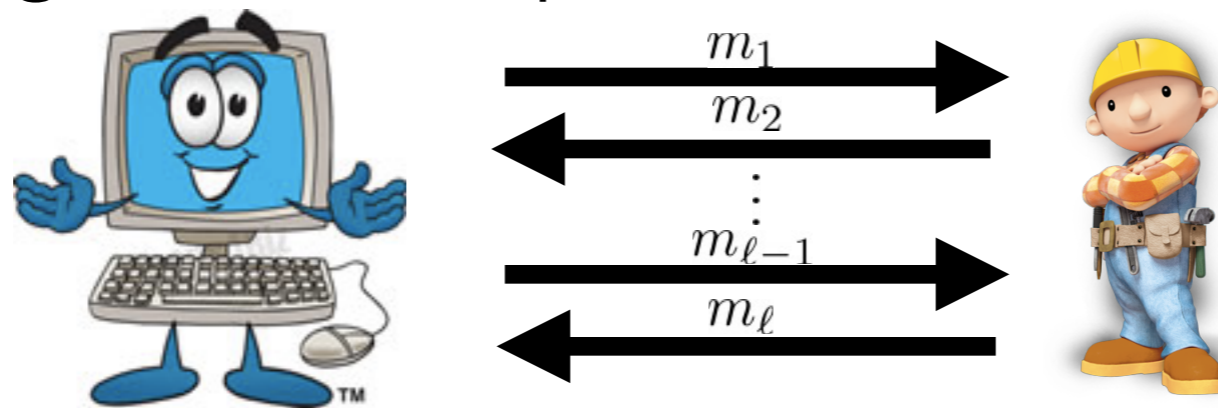
Underlying classical protocol has some functionality.



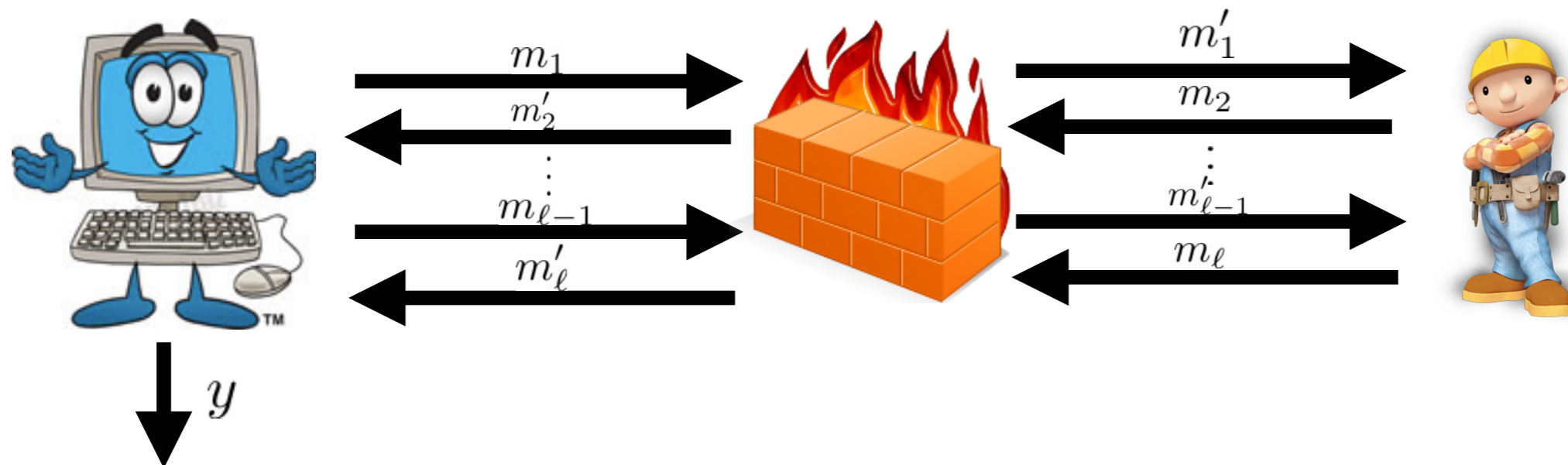
Protocol *with firewall* has same functionality.

# Reverse Firewall Functionality

Underlying classical protocol has some functionality.



Protocol *with firewall* has same functionality.



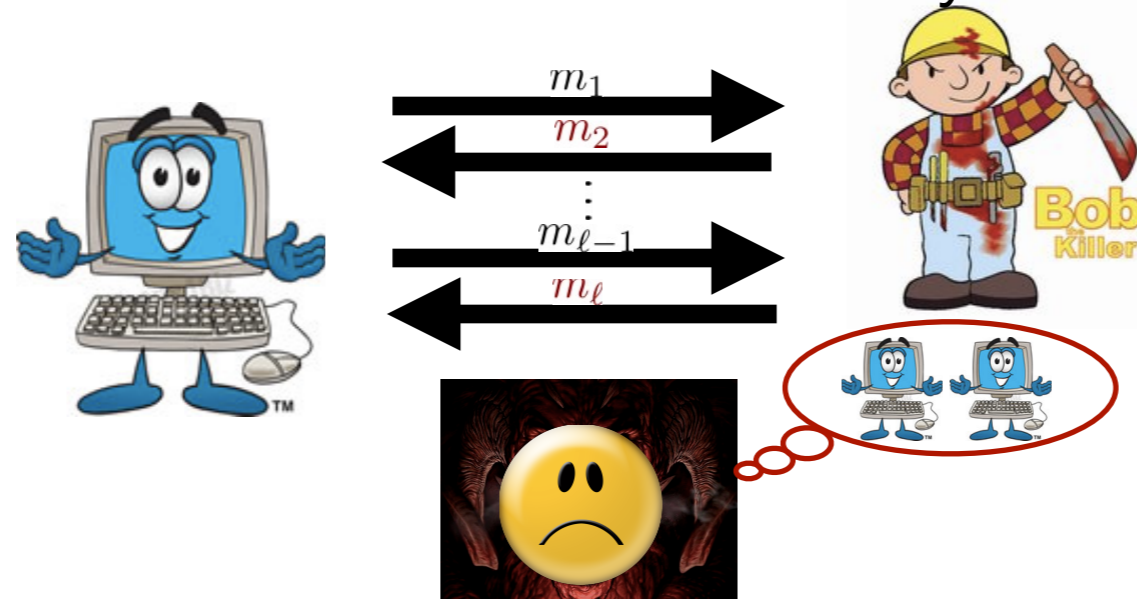
# Reverse Firewall Security

# Reverse Firewall Security

Underlying protocol satisfies some security notion.

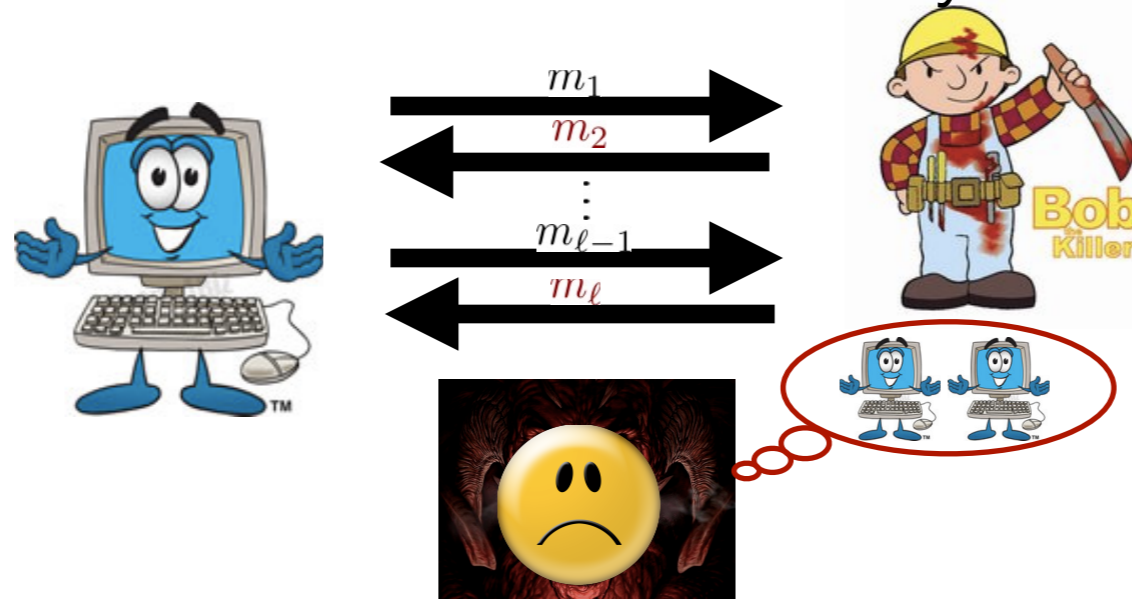
# Reverse Firewall Security


Underlying protocol satisfies some security notion.



# Reverse Firewall Security

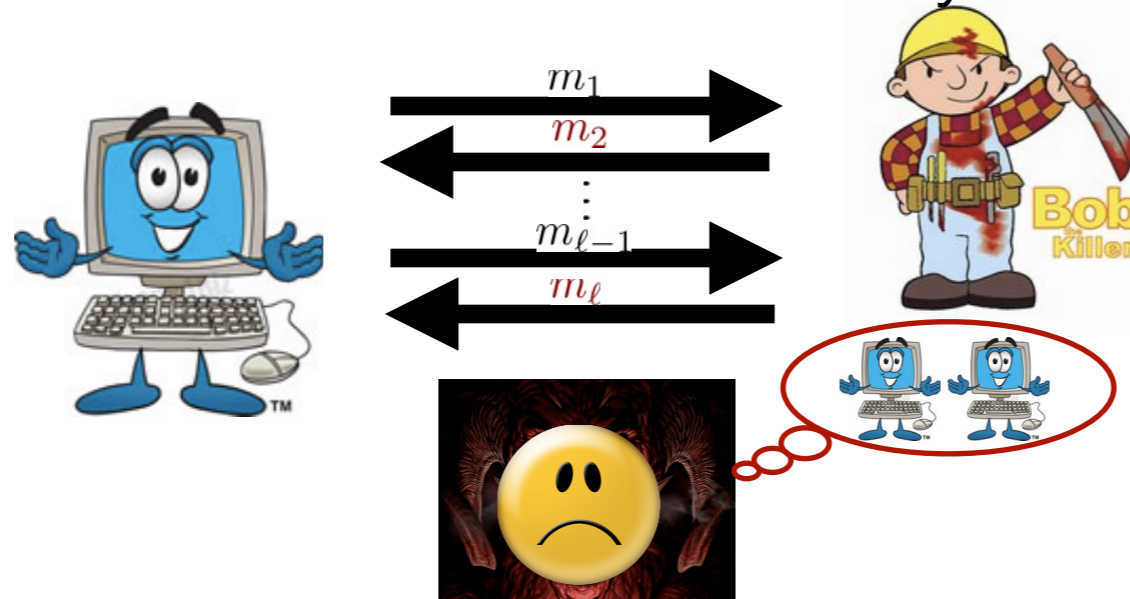
Underlying protocol satisfies some security notion.




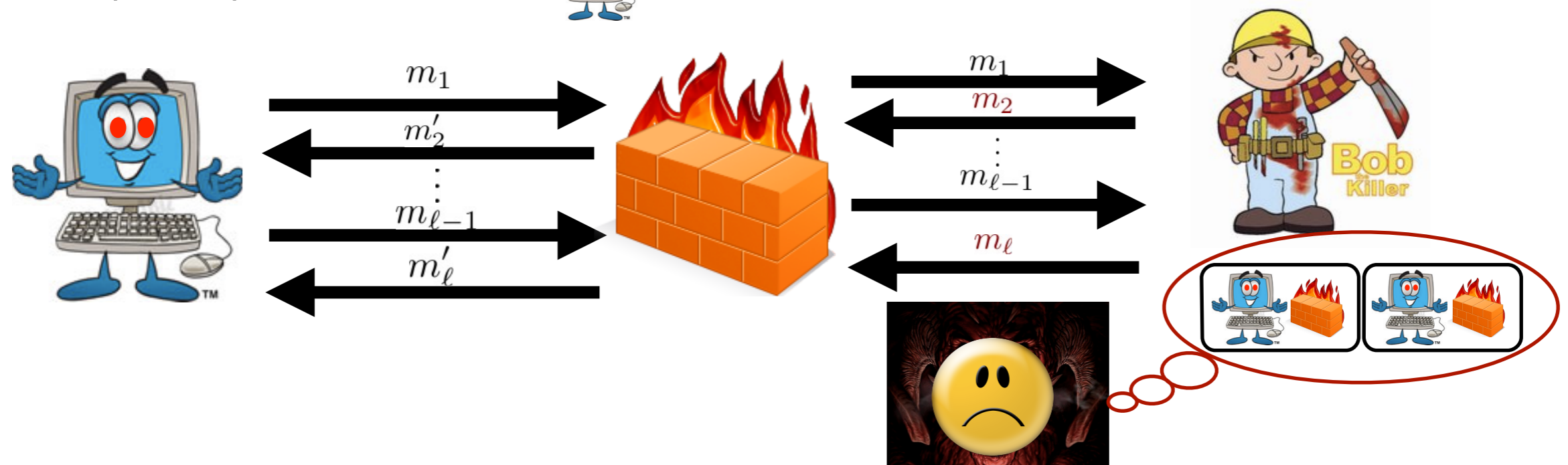
Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*

# Reverse Firewall Security

Underlying protocol satisfies some security notion.

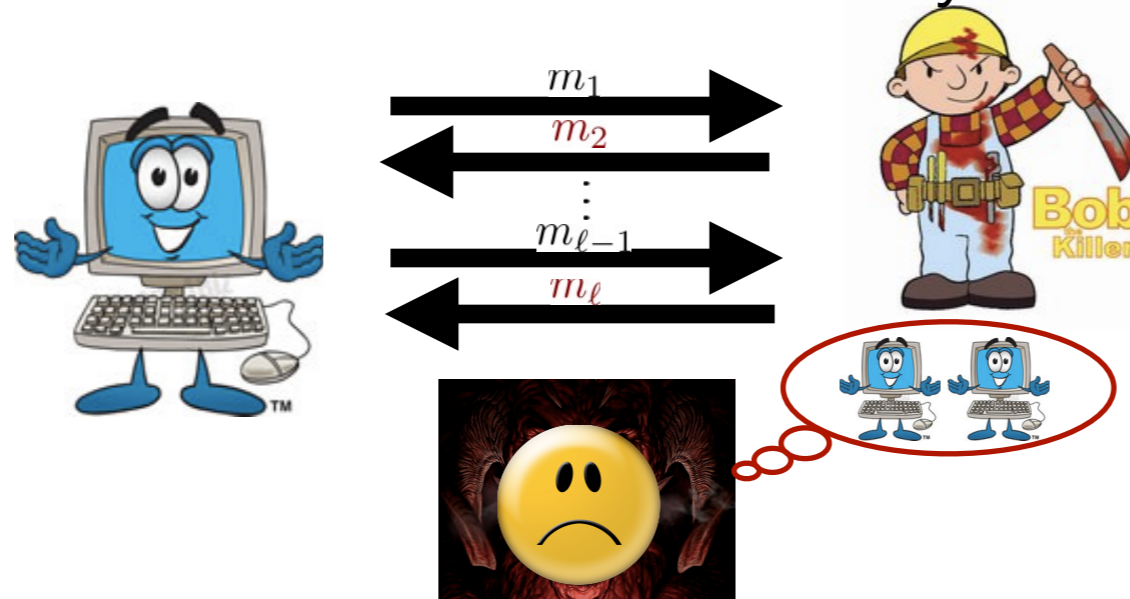



Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*

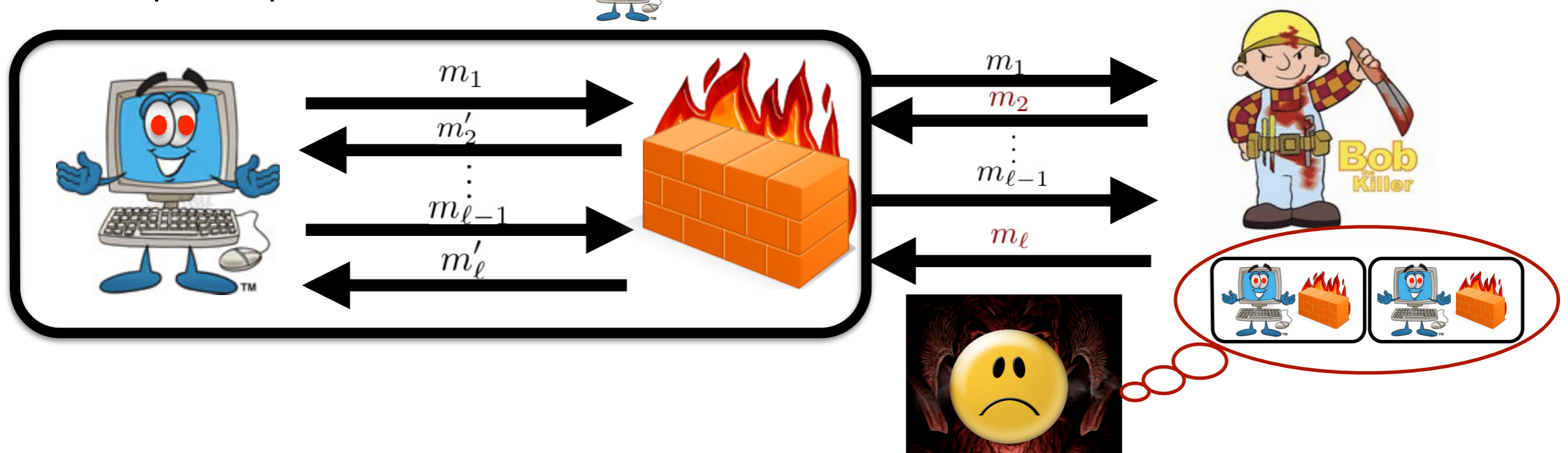


# Reverse Firewall Security

Underlying protocol satisfies some security notion.

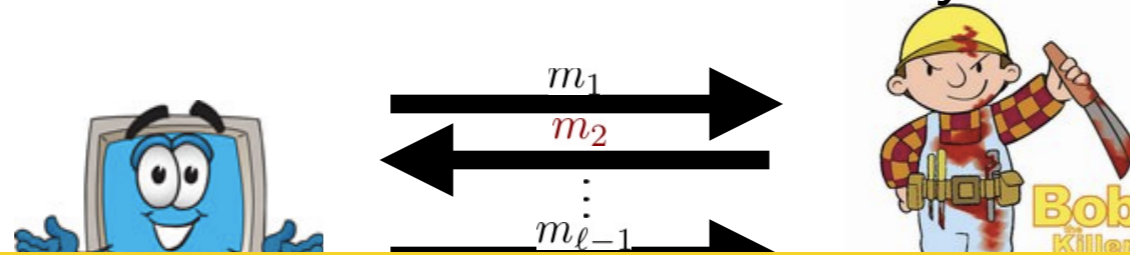


Protocol with firewall satisfies the same security notion *for any efficient corrupt implementation*  *of Alice.*



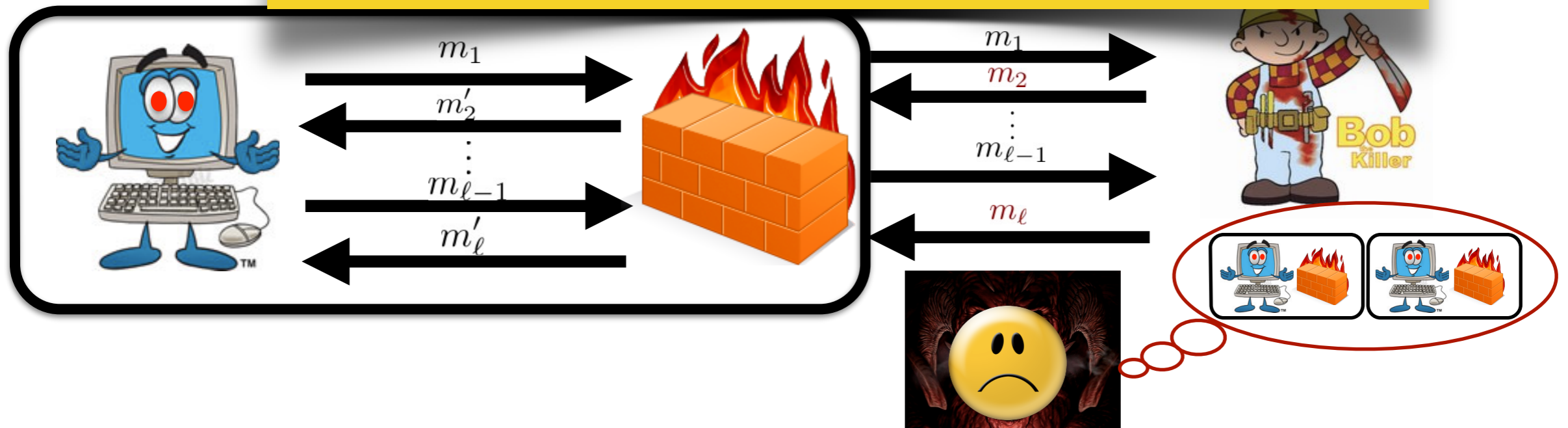
# Reverse Firewall Security

Underlying protocol satisfies some security notion.



**Note:** We require that the protocol is functional and secure *without the reverse firewall* when Alice's implementation is not corrupted.

Protocol w (For “most reasonable security definitions,” this implies *any efficient corrupt im* that “an adversarial firewall can’t break security.”)



# Exfiltration Resistance

# Exfiltration Resistance

A corrupted implementation  of Alice should not be able to leak *any* information to an eavesdropping adversary “through the firewall.”

# Exfiltration Resistance

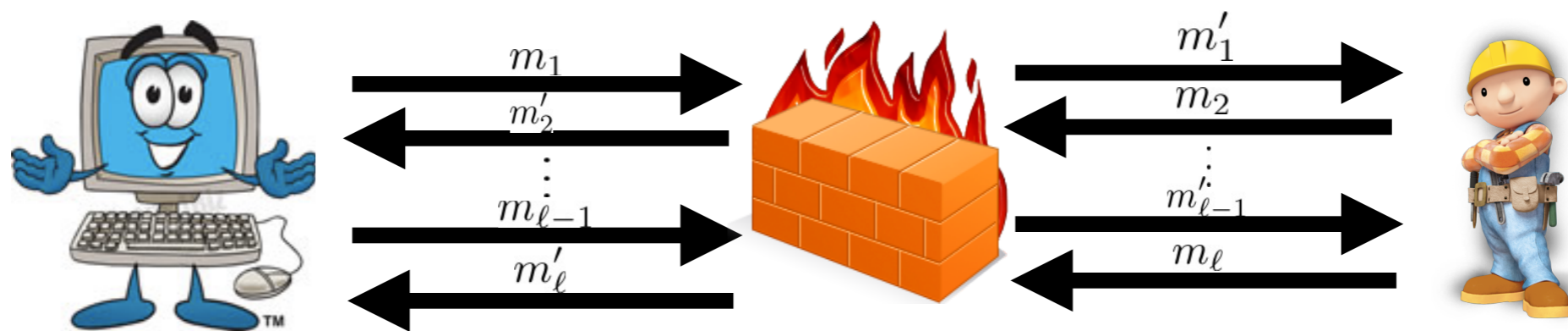
A corrupted implementation  of Alice should not be able to leak *any* information to an eavesdropping adversary “through the firewall.”

(This is often already implied by security of the firewall, but not for all primitives.)

# Exfiltration Resistance

A corrupted implementation  of Alice should not be able to leak *any* information to an eavesdropping adversary “through the firewall.”

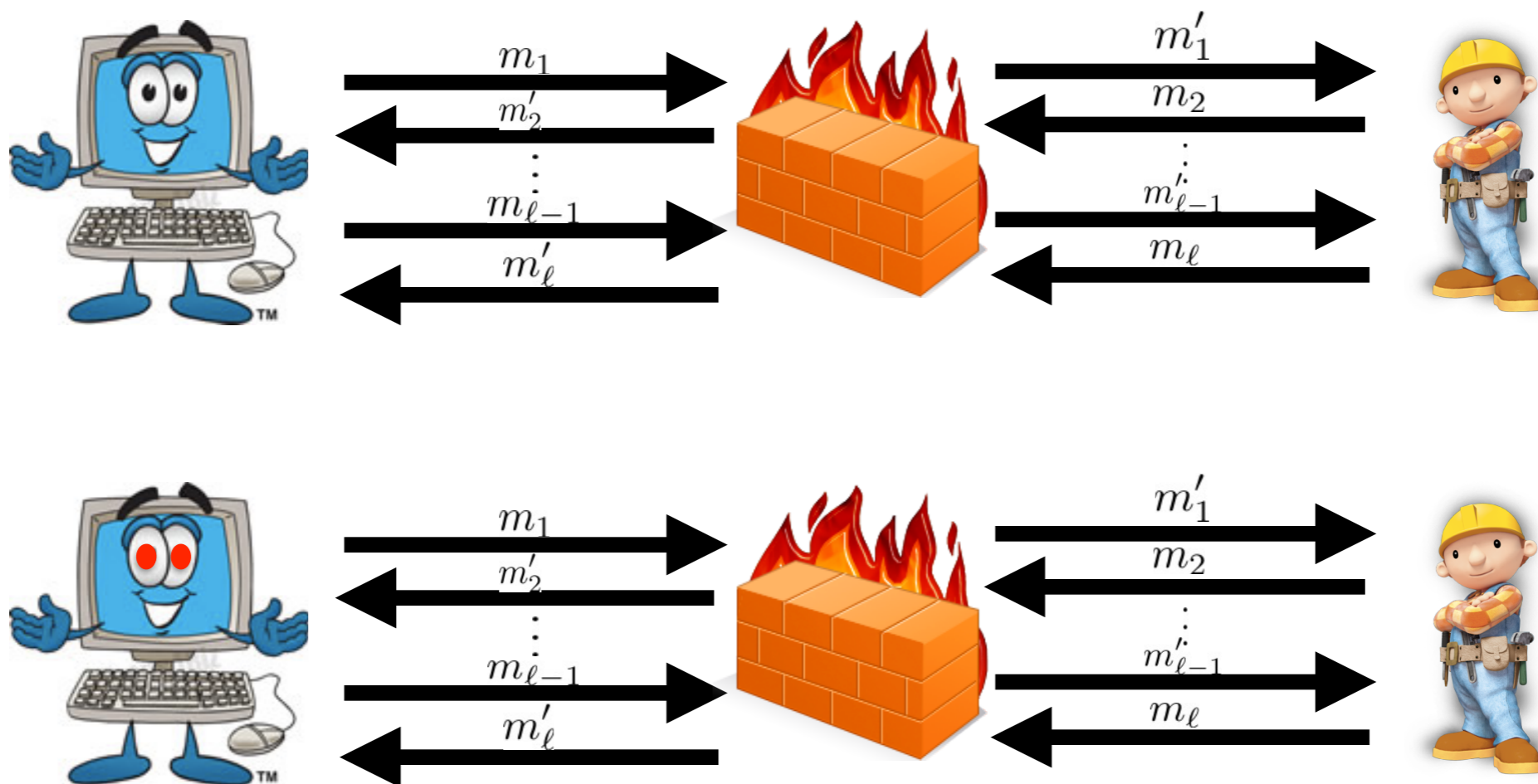
(This is often already implied by security of the firewall, but not for all primitives.)



# Exfiltration Resistance

A corrupted implementation  of Alice should not be able to leak *any* information to an eavesdropping adversary “through the firewall.”

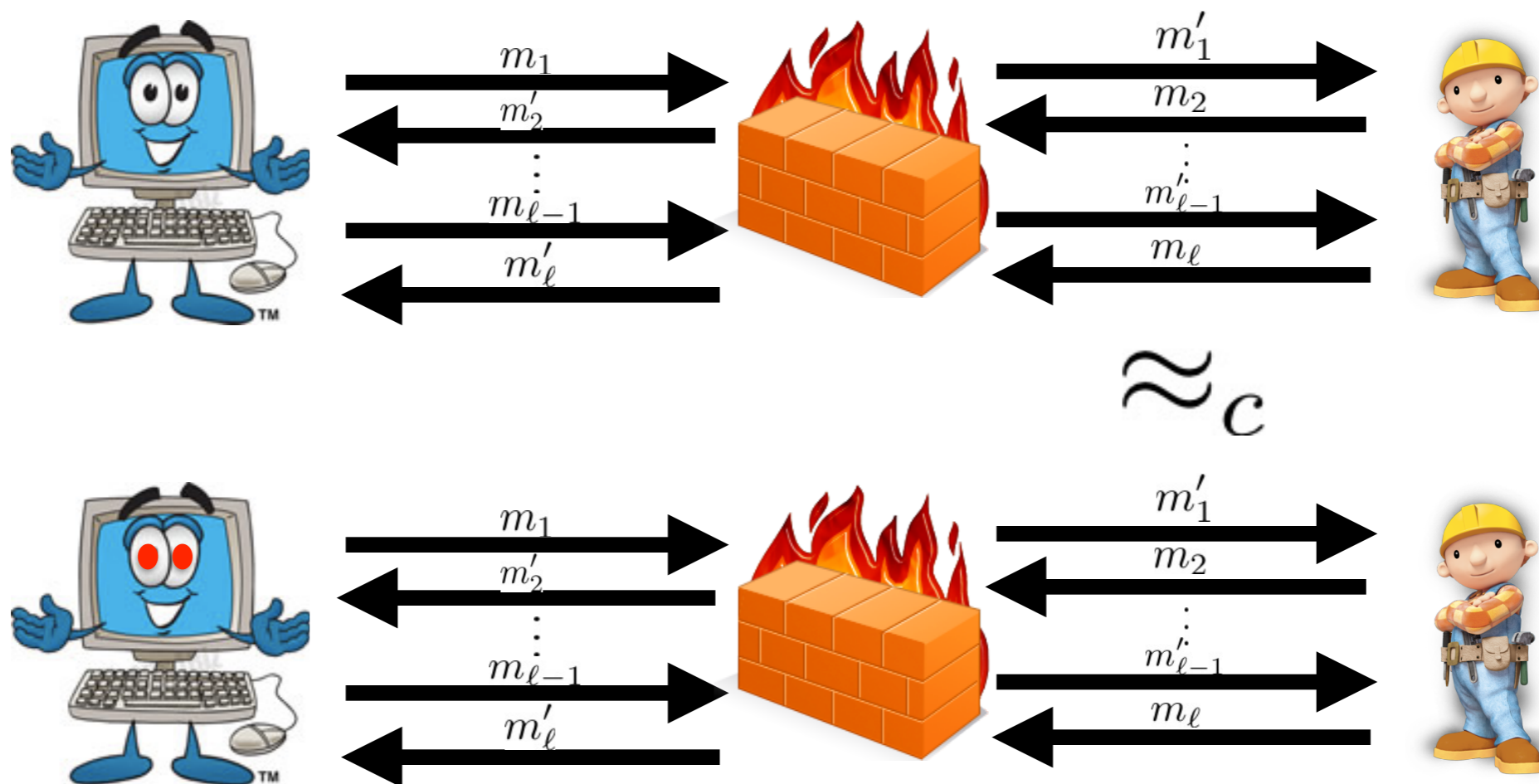
(This is often already implied by security of the firewall, but not for all primitives.)



# Exfiltration Resistance

A corrupted implementation  of Alice should not be able to leak *any* information to an eavesdropping adversary “through the firewall.”

(This is often already implied by security of the firewall, but not for all primitives.)



# Simple Example: Semantically Secure PKE

# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \approx_c \text{Enc}(0)$  for any  $C$ .

# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \approx_c \text{Enc}(0)$  for any  $C$ .

Underlying protocol:

# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \approx_c \text{Enc}(0)$  for any  $C$ .

Underlying protocol:



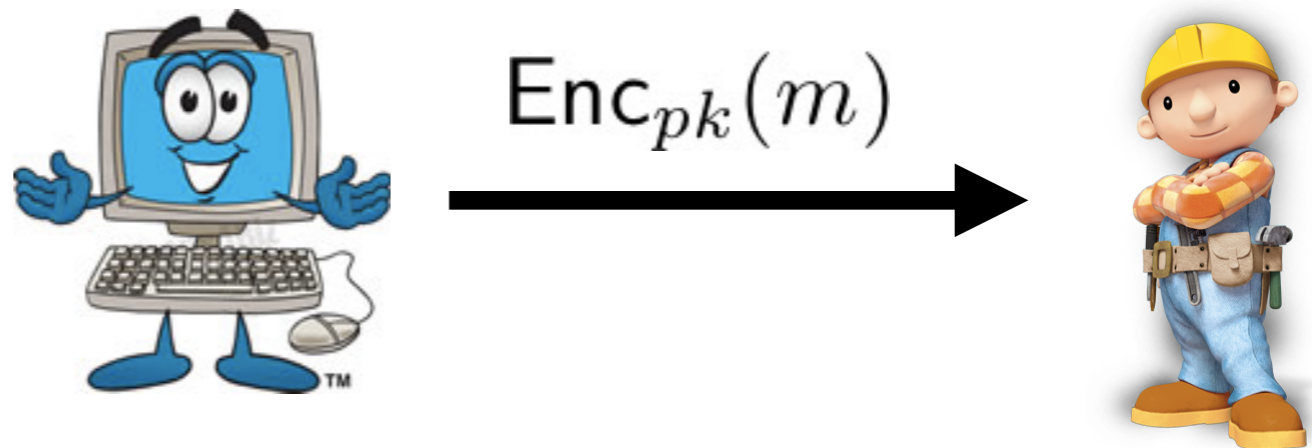
$\text{Enc}_{pk}(m)$



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \approx_c \text{Enc}(0)$  for any  $C$ .

Underlying protocol:

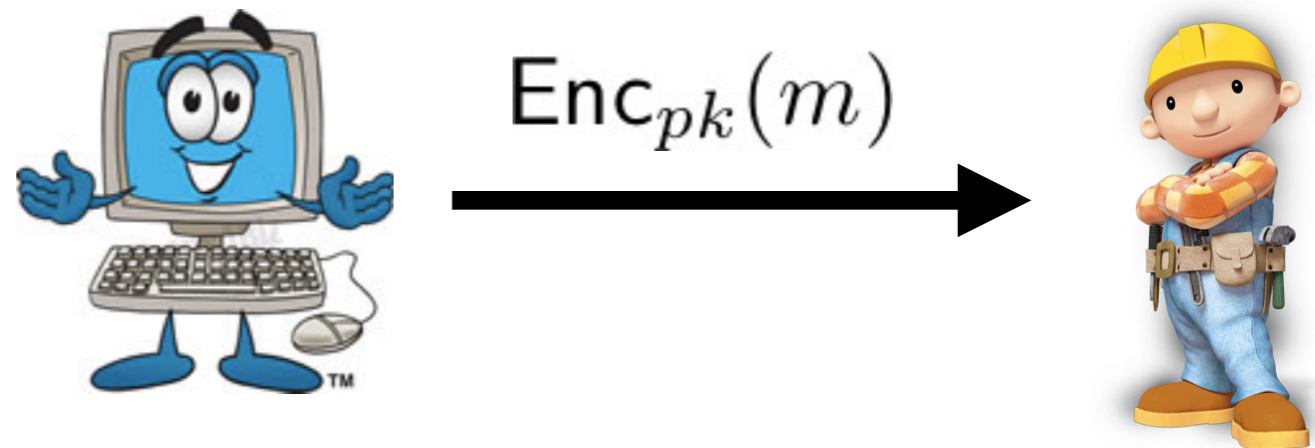


Firewall:

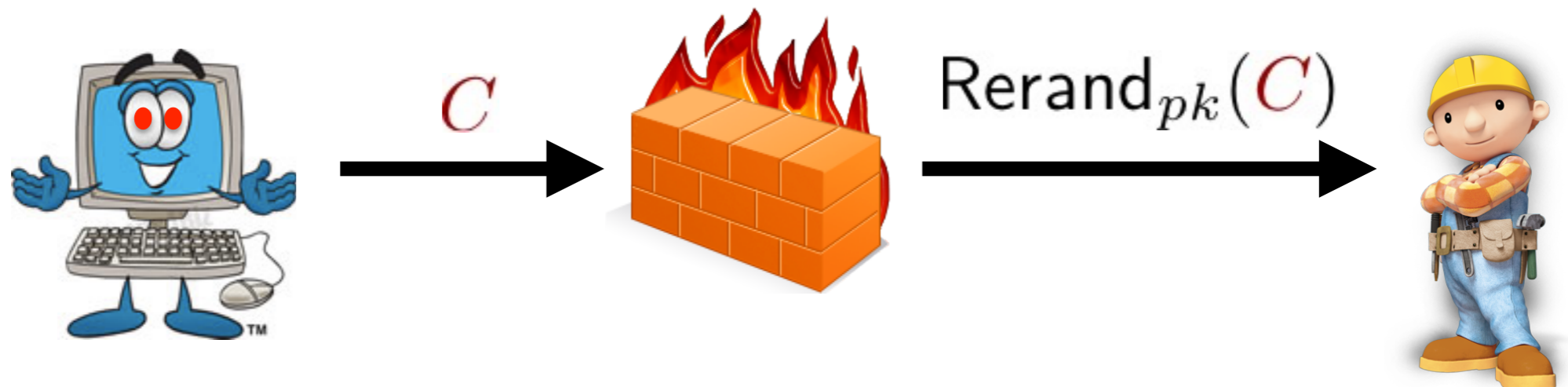
# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \approx_c \text{Enc}(0)$  for any  $C$ .

Underlying protocol:



Firewall:



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

**Functionality:**

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall?

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall?

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall?

Underly

Firewall



# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓

Underly

Firewall




# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure *regardless of how*  *behaves?*

Underly

Firewall




# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure regardless of how  behaves? ✓

Underly

Firewall




# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$

## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure regardless of how  behaves? ✓

## Exfiltration resistance:

Underly

Firewall




# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$




## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure regardless of how  behaves? ✓

## Exfiltration resistance:

- Is  for any  computationally indistinguishable from  ?

Underly

Firewall




# Simple Example: Semantically Secure PKE

Let  $(\text{Enc}, \text{Dec}, \text{Rerand})$  be a *rerandomizable* PKE scheme such that  $\text{Dec}(\text{Rerand}(\text{Enc}(m))) = m$  and  $\text{Rerand}(C) \neq \text{Enc}(0)$



## Functionality:

- Is the protocol functional without the firewall? ✓
- Is the protocol functional with the firewall? ✓

## Security:

- Is the protocol semantically secure without firewall? ✓
- Is the protocol with the firewall semantically secure *regardless of how*  *behaves*? ✓

## Exfiltration resistance:

- Is  for any  computationally ✓  
indistinguishable from  ?

Underly

Firewall



Can we instantiate more  
powerful primitives in this model?

# Secure Function Evaluation

# Secure Function Evaluation



# Secure Function Evaluation



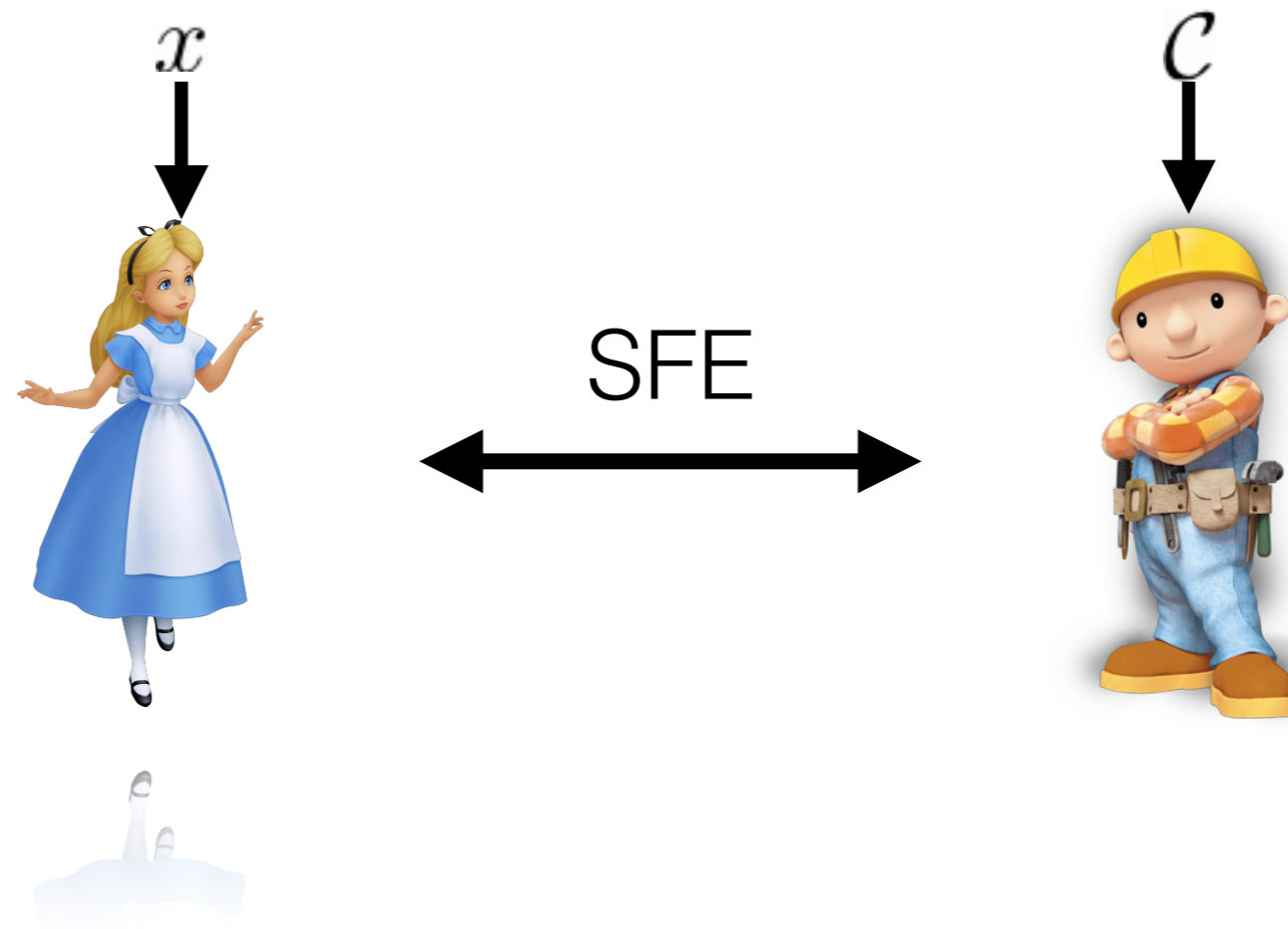
# Secure Function Evaluation



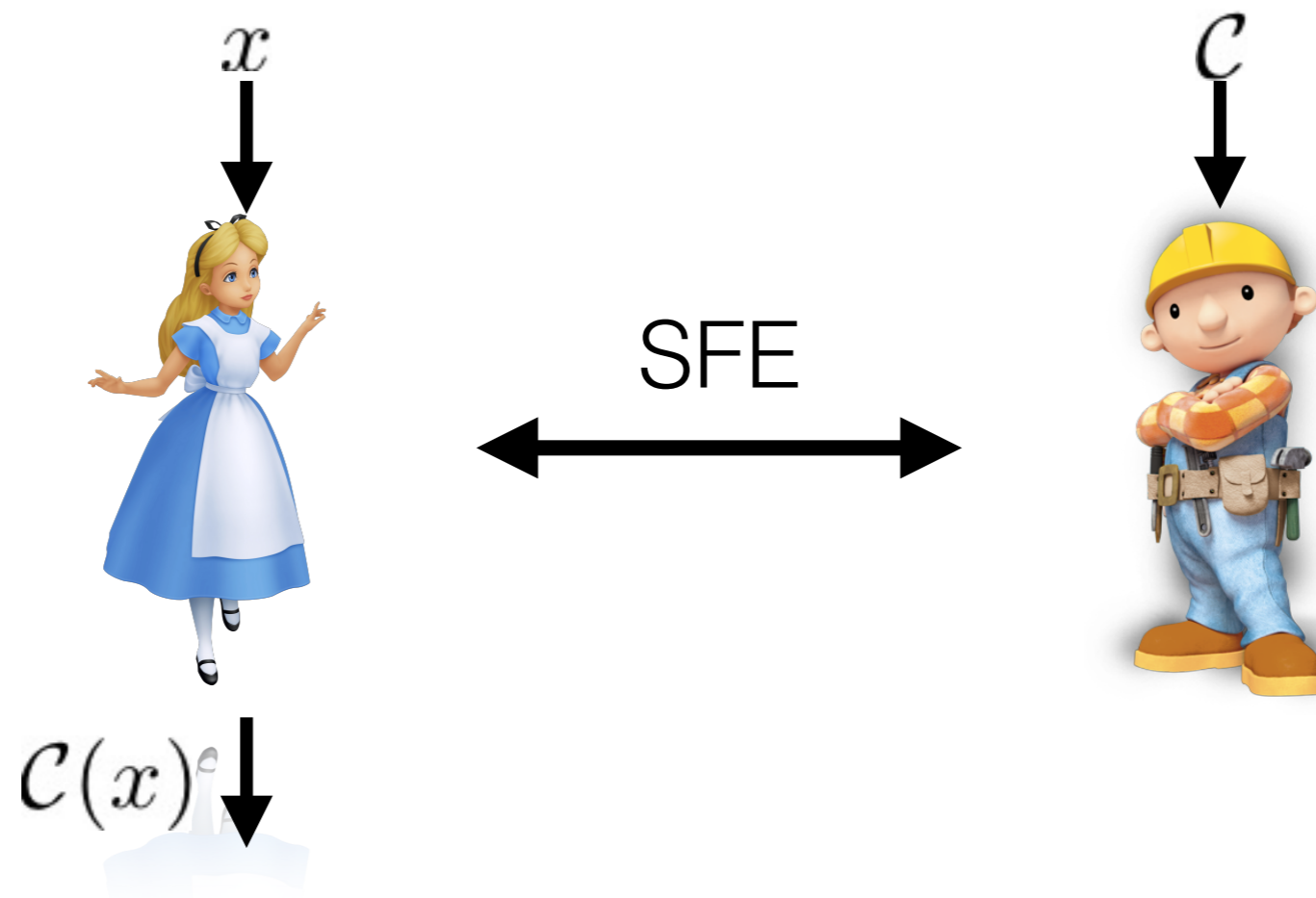
# Secure Function Evaluation



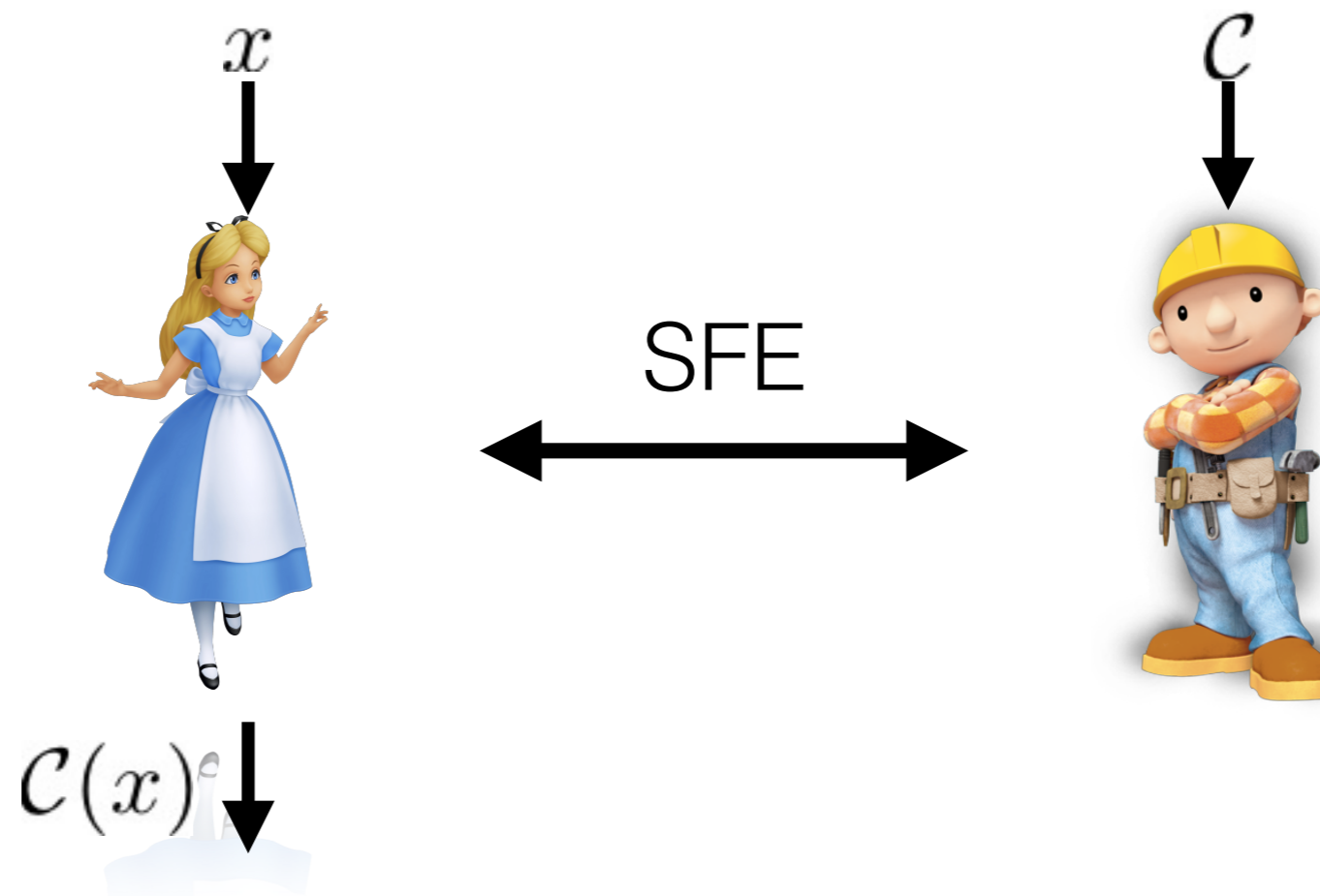
# Secure Function Evaluation



# Secure Function Evaluation

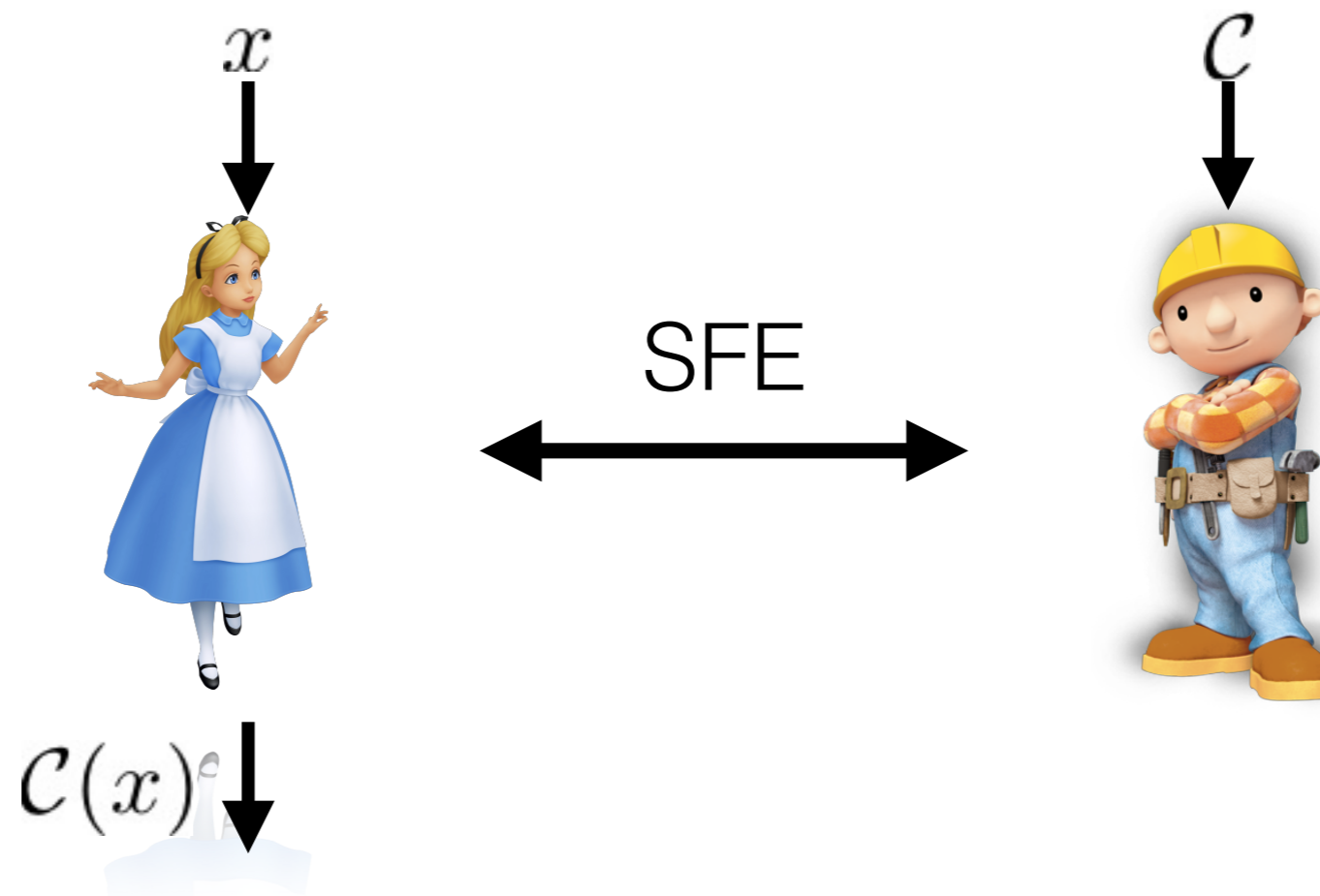


# Secure Function Evaluation



**Security for Alice:** Bob “learns nothing.”

# Secure Function Evaluation



**Security for Alice:** Bob “learns nothing.”

**Security for Bob:** Alice “learns nothing but  $C(x)$  .”

# Secure Function Evaluation with a Reverse Firewall for Each Party

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.
  - New rerandomizable garbled circuit scheme

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.
  - New rerandomizable garbled circuit scheme
    - More efficient and secure than prior constructions [GHV10]

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.
  - New rerandomizable garbled circuit scheme
    - More efficient and secure than prior constructions [GHV10]
- Only two rounds.

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.
  - New rerandomizable garbled circuit scheme
    - More efficient and secure than prior constructions [GHV10]
- Only two rounds.
- Send  $\sim 5$  group elements per gate and compute  $\sim 5$  exponentiations per gate

# Secure Function Evaluation with a Reverse Firewall for Each Party

- Based on Oblivious Transfer and Garbled Circuits [Yao86, BHR12].
  - Naor-Pinkas/Aiello-Ishai-Reingold OT [NP01, AIR01].
    - Rerandomizable and malleable OT.
  - New rerandomizable garbled circuit scheme
    - More efficient and secure than prior constructions [GHV10]
- Only two rounds.
- Send  $\sim 5$  group elements per gate and compute  $\sim 5$  exponentiations per gate
- Security follows from (a slight variant of) DDH.

# Summary

# Summary

- Recent revelations suggest that we can no longer trust our computers when security is paramount.

# Summary

- Recent revelations suggest that we can no longer trust our computers when security is paramount.
- We give a general framework for guaranteeing security of arbitrary cryptographic primitives *even on a compromised machine*.

# Summary

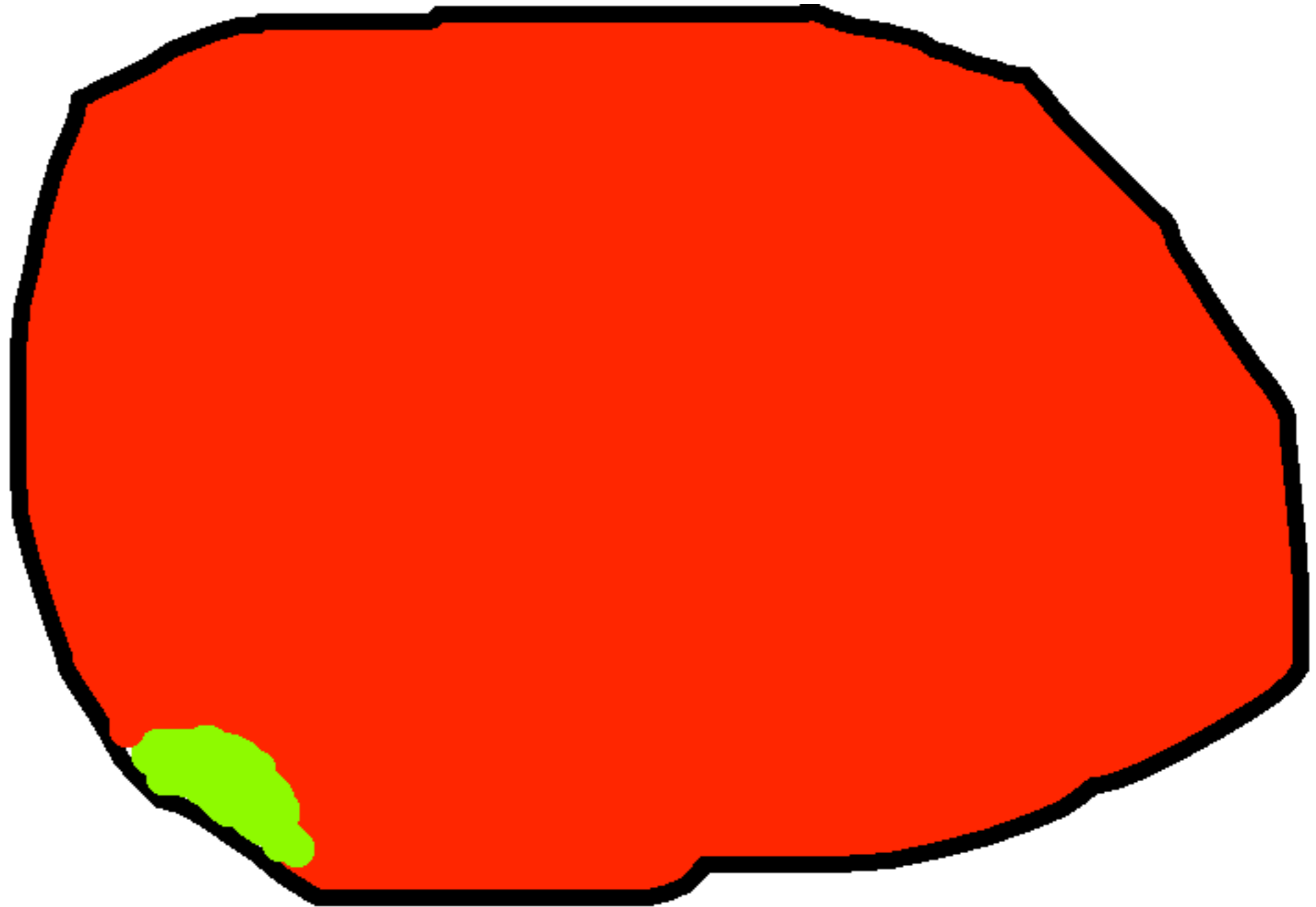
- Recent revelations suggest that we can no longer trust our computers when security is paramount.
- We give a general framework for guaranteeing security of arbitrary cryptographic primitives *even on a compromised machine*.
- We do this without placing any additional trust in a third party.

# Summary

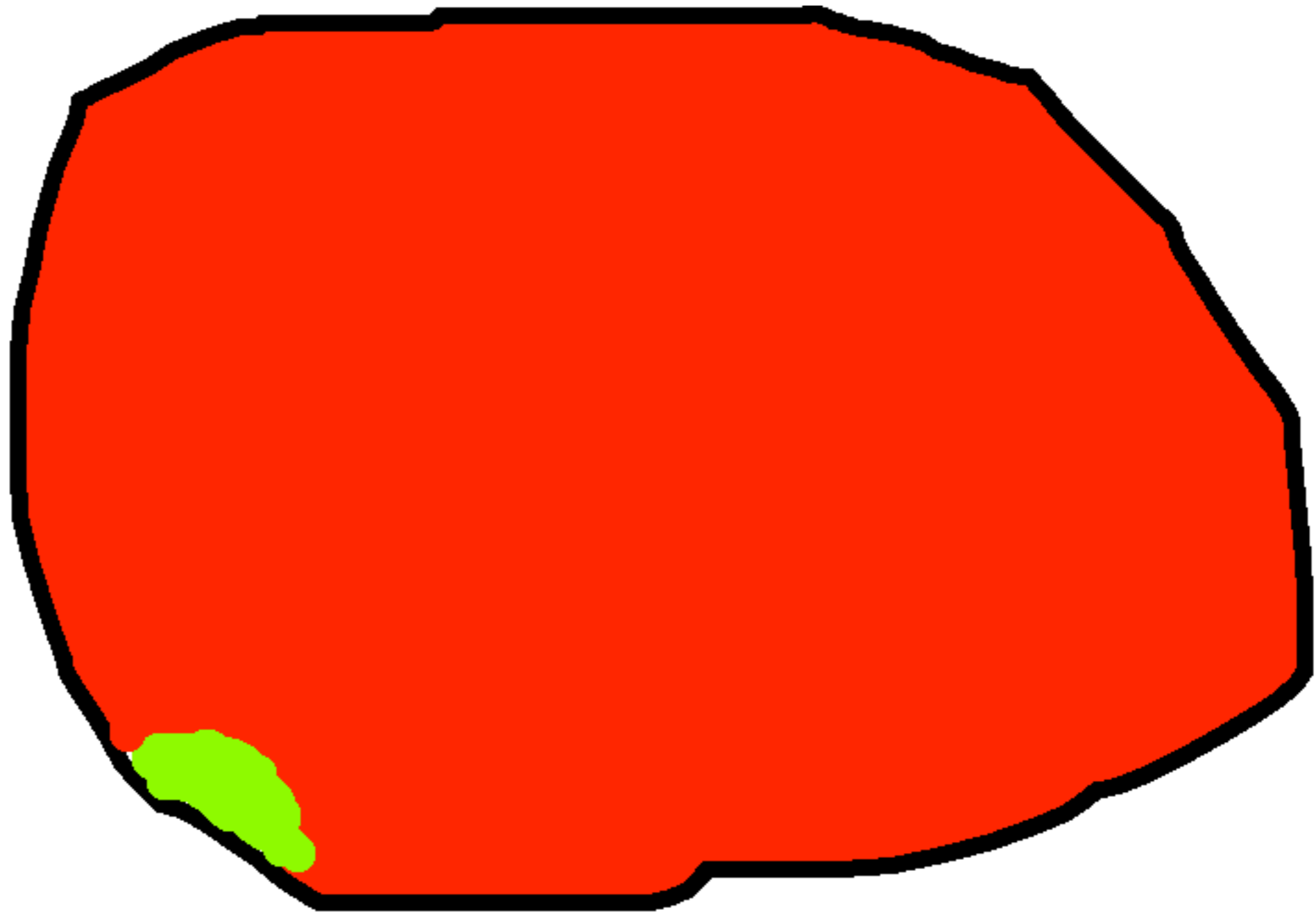
- Recent revelations suggest that we can no longer trust our computers when security is paramount.
- We give a general framework for guaranteeing security of arbitrary cryptographic primitives *even on a compromised machine*.
- We do this without placing any additional trust in a third party.
- We show that we can instantiate very strong cryptographic primitives in this model.

# A Lot of Work to Be Done!

# A Lot of Work to Be Done!



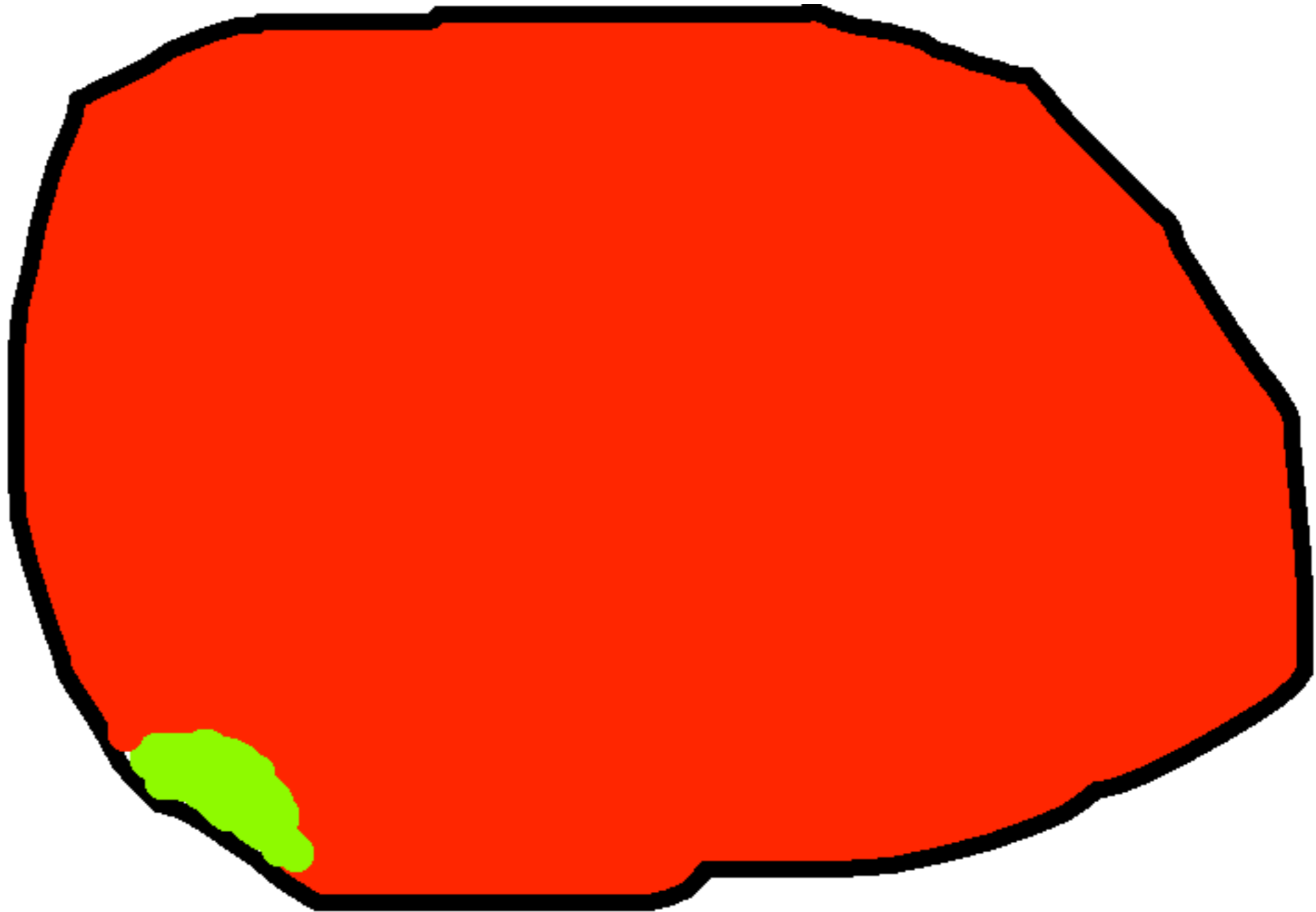
# A Lot of Work to Be Done!



The space of all known cryptographic primitives


# A Lot of Work to Be Done!


 Studied in  
this context

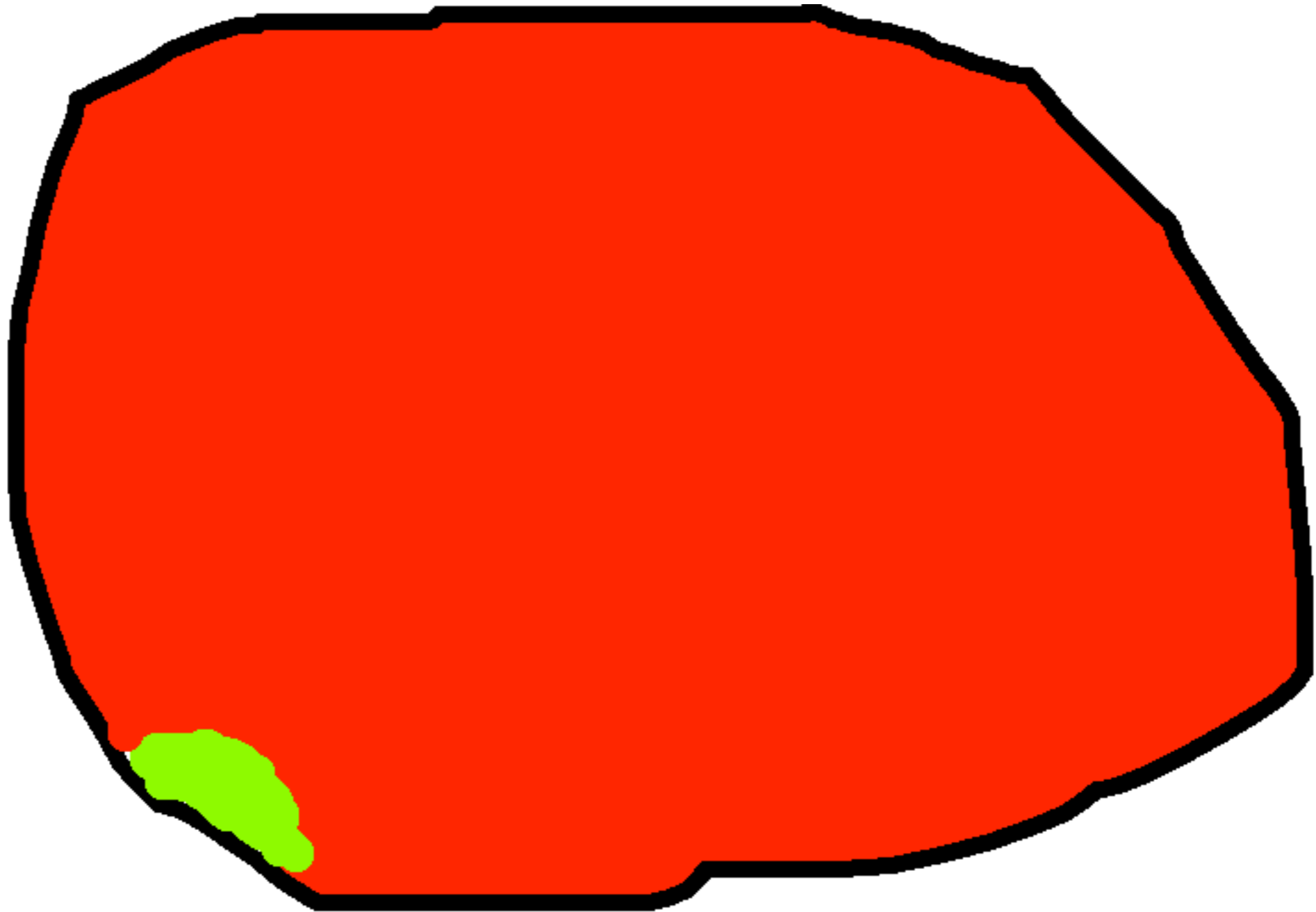


The space of all known cryptographic primitives

# A Lot of Work to Be Done!



 Studied in  
this context

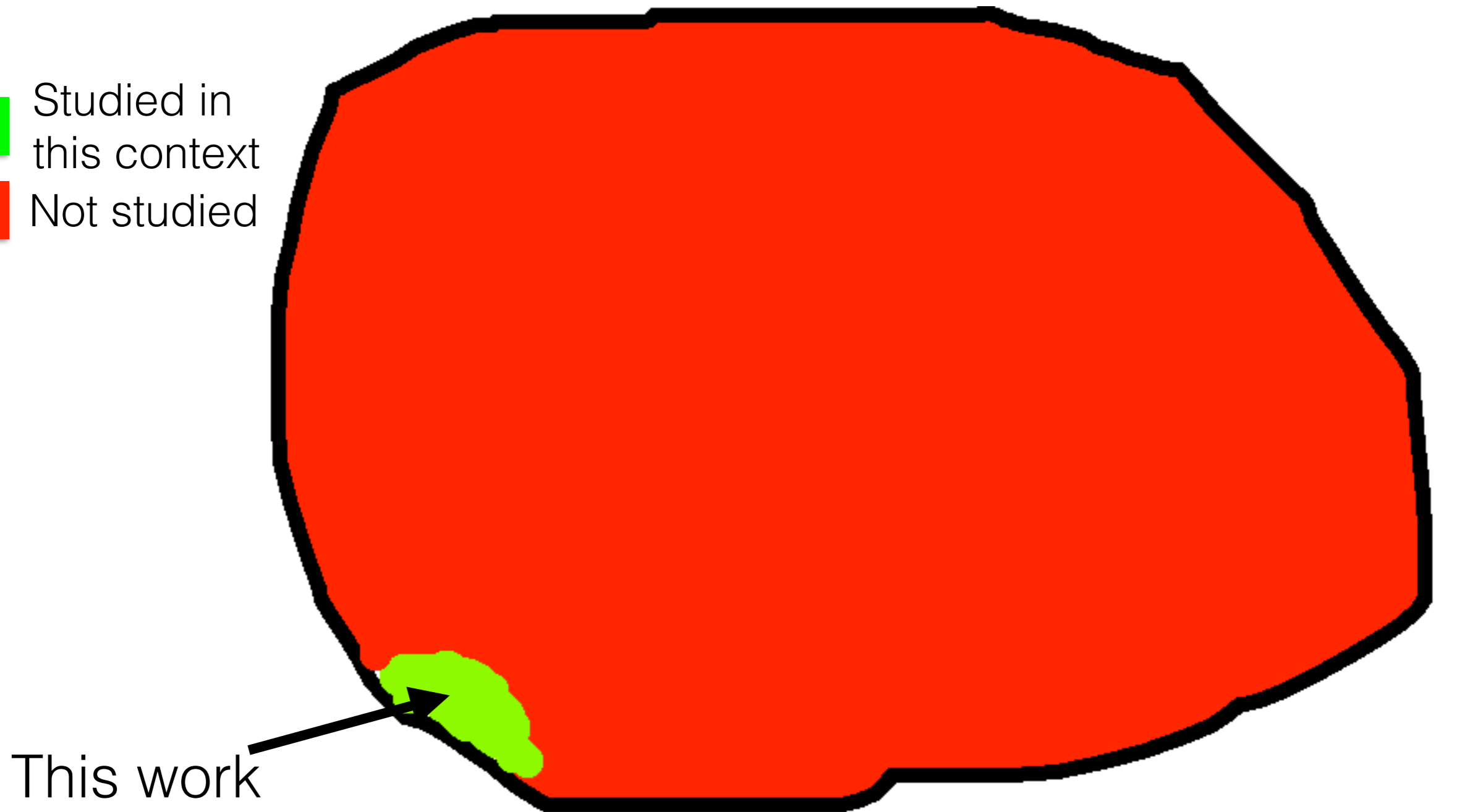
 Not studied



The space of all known cryptographic primitives



# A Lot of Work to Be Done!

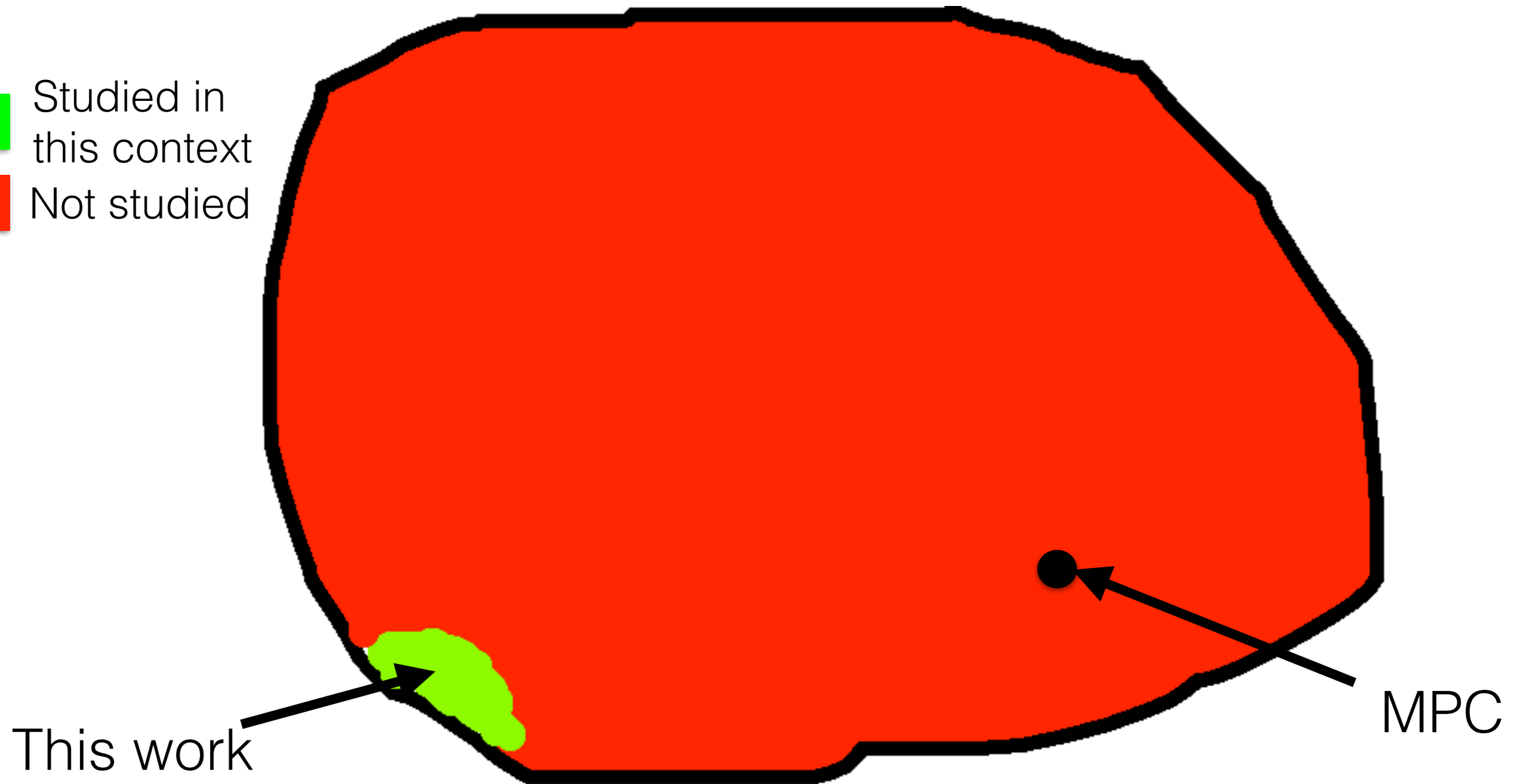
 Studied in  
this context  
 Not studied



The space of all known cryptographic primitives

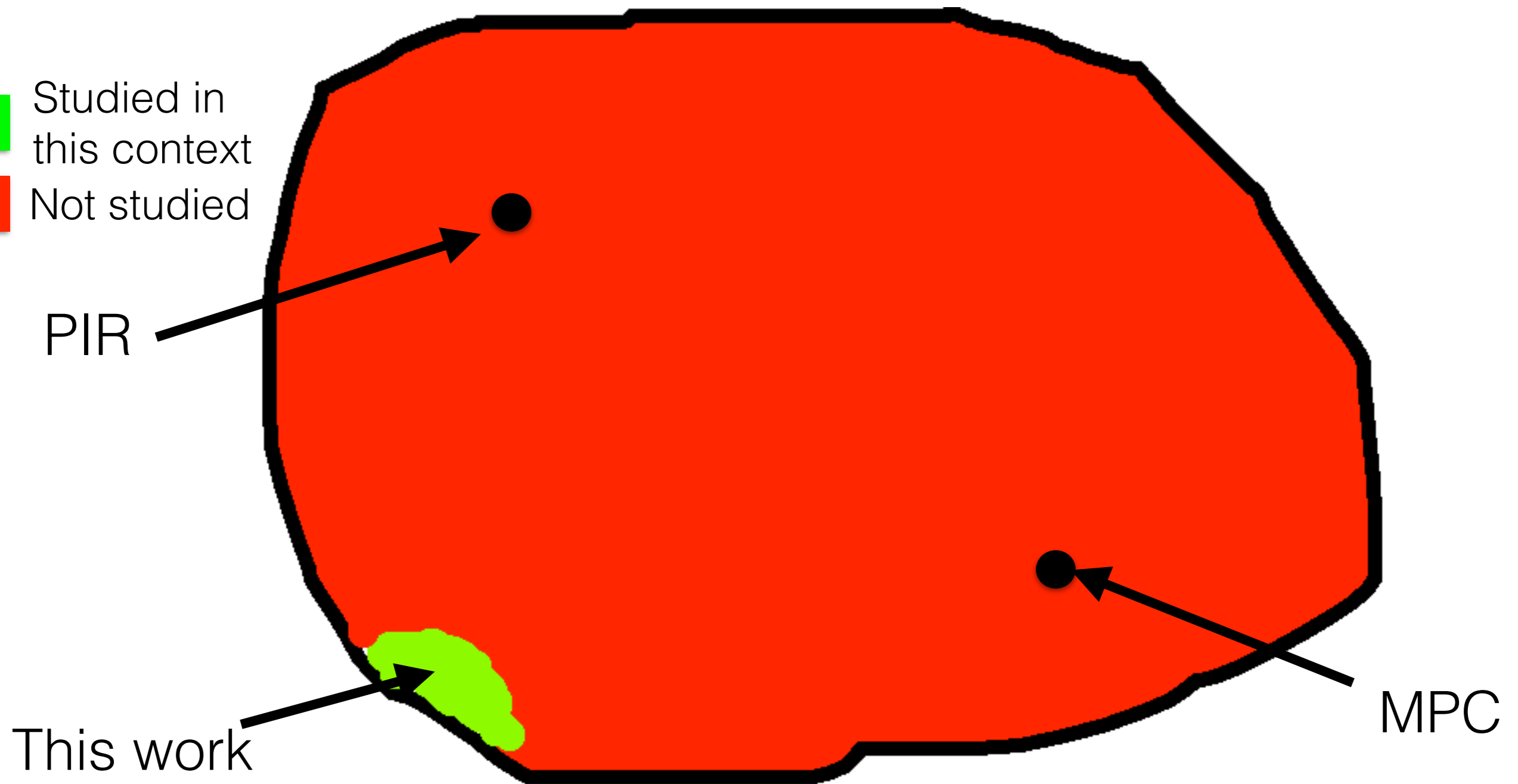
# A Lot of Work to Be Done!

 Studied in  
this context  
 Not studied



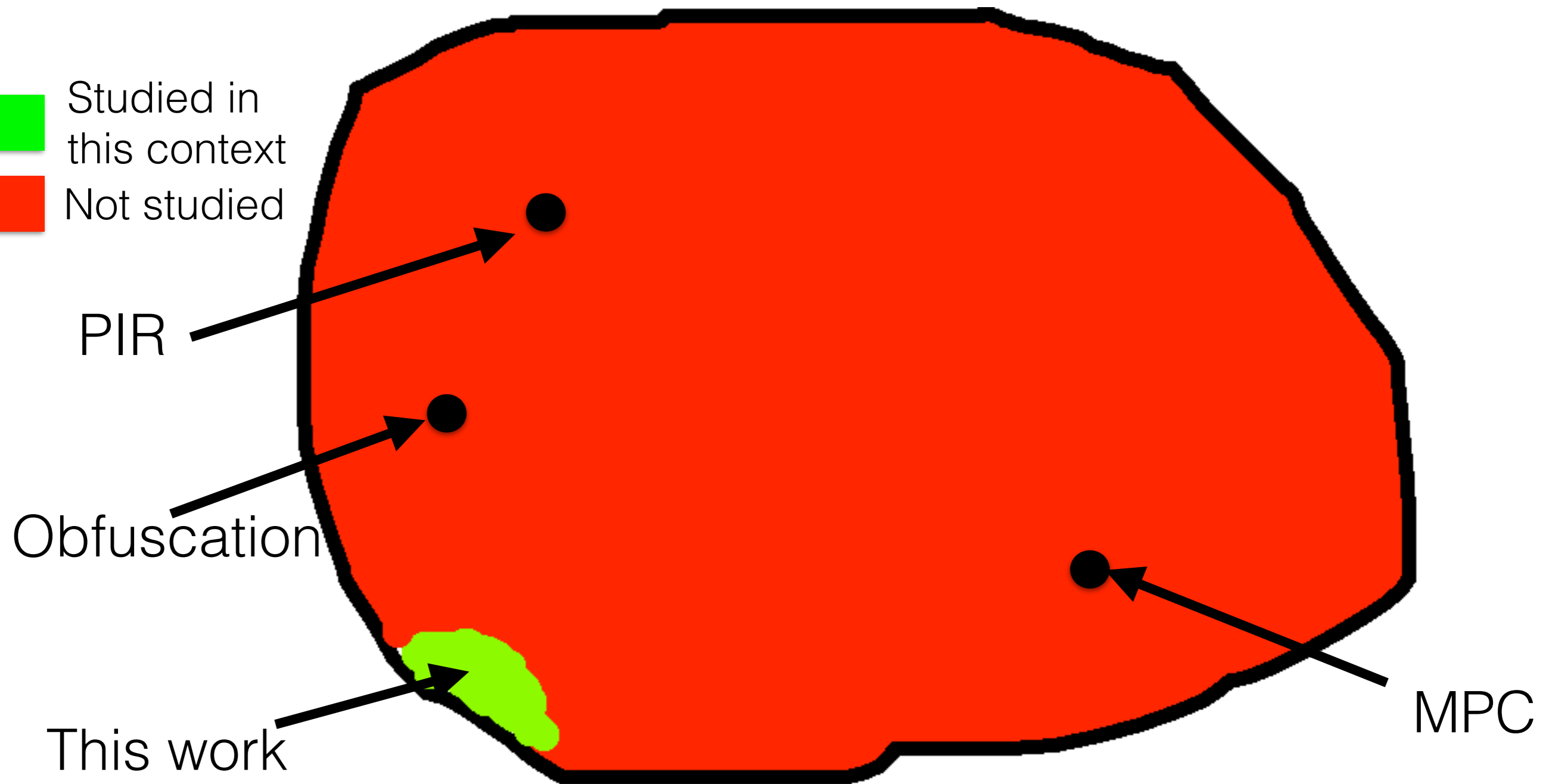
The space of all known cryptographic primitives

# A Lot of Work to Be Done!



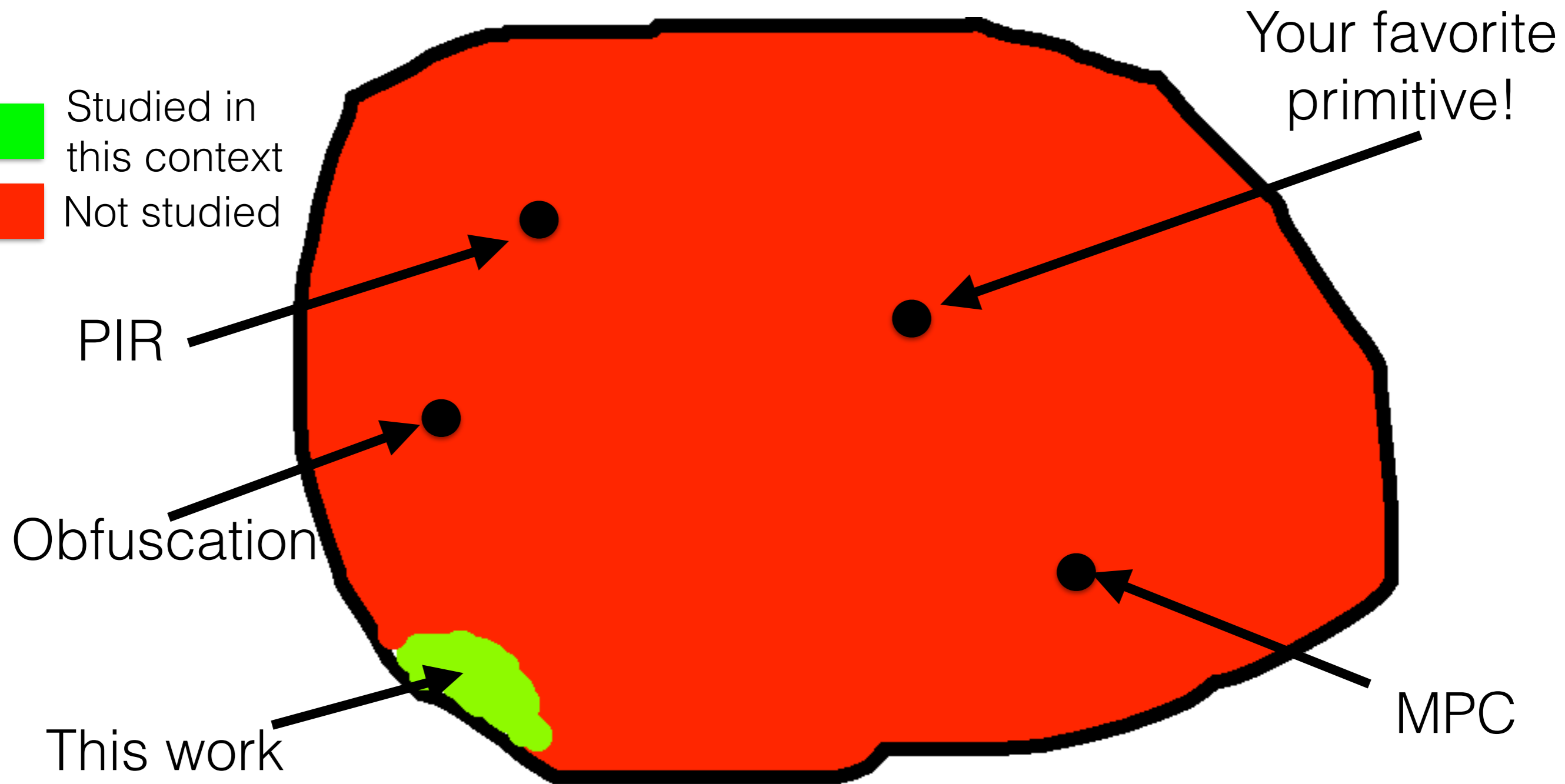
The space of all known cryptographic primitives

# A Lot of Work to Be Done!



The space of all known cryptographic primitives

# A Lot of Work to Be Done!



The space of all known cryptographic primitives

# Thanks!

