BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

McEliece Public Key Cryptosystem

McEliece Public Key Cryptosystem

The **McEliece Public Key Cryptosystem** is based on error-correcting codes.

McEliece Public Key Cryptosystem

The **McEliece Public Key Cryptosystem** is based on error-correcting codes.

1. Named after its inventor, Robert McEliece.

McEliece Public Key Cryptosystem

The **McEliece Public Key Cryptosystem** is based on error-correcting codes.

1. Named after its inventor, Robert McEliece.
2. McEliece public key cryptosystem was published in 1978 but was thought to not be practical because the key is large.

McEliece Public Key Cryptosystem

The **McEliece Public Key Cryptosystem** is based on error-correcting codes.

1. Named after its inventor, Robert McEliece.
2. McEliece public key cryptosystem was published in 1978 but was thought to not be practical because the key is large.
3. McEliece public key cryptosystem is getting more attention now since (1) it's not based on factoring or other number theory assumptions, (2) we are able to handle bigger keys now, and (3) there are some applications where key size can be big, for example, if you only generate the key once a month.

McEliece Public Key Cryptosystem

The **McEliece Public Key Cryptosystem** is based on error-correcting codes.

1. Named after its inventor, Robert McEliece.
2. McEliece public key cryptosystem was published in 1978 but was thought to not be practical because the key is large.
3. McEliece public key cryptosystem is getting more attention now since (1) it's not based on factoring or other number theory assumptions, (2) we are able to handle bigger keys now, and (3) there are some applications where key size can be big, for example, if you only generate the key once a month.
4. McEliece public key cryptosystem is a candidate for NIST's quantum-resistant public key challenge.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Recall that with DH and RSA we
spent X slides on Number theory and 1 on the protocol.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Recall that with DH and RSA we
spent X slides on Number theory and 1 on the protocol.

Recall that with LWE we
spent Y slides on Linear Algebra and 1 on the protocol.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Recall that with DH and RSA we
spent X slides on Number theory and 1 on the protocol.

Recall that with LWE we
spent Y slides on Linear Algebra and 1 on the protocol.

Similarly, for McEliece we will
spend Z slides on Error Corr. Codes and 1 on the protocol.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Recall that with DH and RSA we
spent X slides on Number theory and 1 on the protocol.

Recall that with LWE we
spent Y slides on Linear Algebra and 1 on the protocol.

Similarly, for McEliece we will
spend Z slides on Error Corr. Codes and 1 on the protocol.

1. Modern Crypto is able to draw upon math already known.

Math Needed for Various Protocols

For this slide X,Y,Z are between 40 and 80.

Recall that with DH and RSA we
spent X slides on Number theory and 1 on the protocol.

Recall that with LWE we
spent Y slides on Linear Algebra and 1 on the protocol.

Similarly, for McEliece we will
spend Z slides on Error Corr. Codes and 1 on the protocol.

1. Modern Crypto is able to draw upon math already known.
2. Many protocols use elementary math since complicated math might be harder to code up and may have larger constants.

A Long Aside:
Error Correcting Codes

Intentional Error Detection in Real Life

In Sept I emailed my TA's

Reminder: TA meeting Thursday Sept 16 at 8:30PM

Intentional Error Detection in Real Life

In Sept I emailed my TA's

Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

Intentional Error Detection in Real Life

In Sept I emailed my TA's
Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

This is an error check. If the date is NOT that day of the week then they will recognize that I made an error and email me.

Intentional Error Detection in Real Life

In Sept I emailed my TA's
Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

This is an error check. If the date is NOT that day of the week then they will recognize that I made an error and email me.

It worked Josh emailed me

Intentional Error Detection in Real Life

In Sept I emailed my TA's
Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

This is an error check. If the date is NOT that day of the week then they will recognize that I made an error and email me.

It worked Josh emailed me
Bill you moron, Sept 16 is not a Thursday

Intentional Error Detection in Real Life

In Sept I emailed my TA's
Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

This is an error check. If the date is NOT that day of the week then they will recognize that I made an error and email me.

It worked Josh emailed me
Bill you moron, Sept 16 is not a Thursday
I then checked my calendar and emailed out the correct date.

Intentional Error Detection in Real Life

In Sept I emailed my TA's
Reminder: TA meeting Thursday Sept 16 at 8:30PM

Why did I include both the day of the week (Thursday) and the date (Sept 16)?

This is an error check. If the date is NOT that day of the week then they will recognize that I made an error and email me.

It worked Josh emailed me
Bill you moron, Sept 16 is not a Thursday
I then checked my calendar and emailed out the correct date.

This is a real-world example of intentional error **detection.**

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

1. Spammers think **Bill will pick up thinking it's neighbor.**

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

1. Spammers think **Bill will pick up thinking it's neighbor.**
2. Bill thinks **DO NOT pick up—it begins (301) 781, so it's a spammer.**

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

1. Spammers think **Bill will pick up thinking it's neighbor.**
2. Bill thinks **DO NOT pick up—it begins (301) 781, so it's a spammer.**

Even stranger: they also do this trick on my cell phone, for which the prefix has nothing to do with geography.

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

1. Spammers think **Bill will pick up thinking it's neighbor.**
2. Bill thinks **DO NOT pick up—it begins (301) 781, so it's a spammer.**

Even stranger: they also do this trick on my cell phone, for which the prefix has nothing to do with geography.

Unintentional Error Detection If the prefix is (301) 781 then I detect that it's a spam call. Unintentional on spammers part.

Unintentional Error Detection in Real Life

My landline number is (301) 781-XXXX. I have caller ID.

Spammers want me to pick up so they will call from a number that begins (301) 781.

1. Spammers think **Bill will pick up thinking it's neighbor.**
2. Bill thinks **DO NOT pick up—it begins (301) 781, so it's a spammer.**

Even stranger: they also do this trick on my cell phone, for which the prefix has nothing to do with geography.

Unintentional Error Detection If the prefix is (301) 781 then I detect that it's a spam call. Unintentional on spammers part.

Another example: The term **Urgent** in the subject line of an email means **this is spam you can ignore**.

whp means with High Probability

whp means **w**ith **h**igh **p**robability. High enough that we will not worry about it not happening.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

The extra bits will help detect or correct errors.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

The extra bits will help detect or correct errors.

This is **not** crypto. There is no Eve.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

The extra bits will help detect or correct errors.

This is **not** crypto. There is no Eve.

Error-correcting means Bob discovers there is an error and where it is, so he can correct it.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

The extra bits will help detect or correct errors.

This is **not** crypto. There is no Eve.

Error-correcting means Bob discovers there is an error and where it is, so he can correct it.

Everything is mod 2.

Scenario and Conventions

Alice and Bob are communicating over a noisy channel.
Alice wants to send Message $m_1 \cdots m_k$. She will send $b_1 \cdots b_n$ where $n > k$.

The extra bits will help detect or correct errors.

This is **not** crypto. There is no Eve.

Error-correcting means Bob discovers there is an error and where it is, so he can correct it.

Everything is mod 2.

A **code** is a map $\{0,1\}^k$ to $\{0,1\}^n$ for error corr. or det.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0 + 1 + 1 + 0 \equiv 0$. He is confident he got the msg. He did.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0 + 1 + 1 + 0 \equiv 0$. He is confident he got the msg. He did.
2. Bob receives 01110, notes $0 + 1 + 1 + 1 \not\equiv 0$. He knows there is an error, but not where it is.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0 + 1 + 1 + 0 \equiv 0$. He is confident he got the msg. He did.
2. Bob receives 01110, notes $0 + 1 + 1 + 1 \not\equiv 0$. He knows there is an error, but not where it is.
3. Bob receives 01101, notes $0 + 1 + 1 + 0 \not\equiv 1$. He knows there is an error, but not where it is.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1b_2b_3b_4$ send $b_1b_2b_3b_4(\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0+1+1+0 \equiv 0$. He is confident he got the msg. He did.
2. Bob receives 01110, notes $0+1+1+1 \not\equiv 0$. He knows there is an error, but not where it is.
3. Bob receives 01101, notes $0+1+1+0 \not\equiv 1$. He knows there is an error, but not where it is.
4. Bob receives 00000, notes $0+0+0+0 \equiv 0$. He is confident he got the msg. He is wrong.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.
Example Parity Check.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.
Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0+1+1+0 \equiv 0$. He is confident he got the msg. He did.
2. Bob receives 01110, notes $0+1+1+1 \not\equiv 0$. He knows there is an error, but not where it is.
3. Bob receives 01101, notes $0+1+1+0 \not\equiv 1$. He knows there is an error, but not where it is.
4. Bob receives 00000, notes $0+0+0+0 \equiv 0$. He is confident he got the msg. He is wrong.

Parity check detects 1 error but not 2. Detect 2 errors: HW.

Error Detecting Codes

Error is detected whp. **Do not know** where error is.

Example Parity Check.

To send $b_1 b_2 b_3 b_4$ send $b_1 b_2 b_3 b_4 (\sum_{i=1}^{4} b_i \pmod 2)$.

Example Alice wants to send 0110. So she sends 01100.

1. Bob receives 01100, notes $0 + 1 + 1 + 0 \equiv 0$. He is confident he got the msg. He did.
2. Bob receives 01110, notes $0 + 1 + 1 + 1 \not\equiv 0$. He knows there is an error, but not where it is.
3. Bob receives 01101, notes $0 + 1 + 1 + 0 \not\equiv 1$. He knows there is an error, but not where it is.
4. Bob receives 00000, notes $0 + 0 + 0 + 0 \equiv 0$. He is confident he got the msg. He is wrong.

Parity check detects 1 error but not 2. Detect 2 errors: HW.

We will NOT use Error Detection for McEliece Cipher.

Error Correction Codes

Error is detected whp. **Do know** where error is.

Error Correction Codes

Error is detected whp. **Do know** where error is.

Example Repetition Code.

Error Correction Codes

Error is detected whp. **Do know** where error is.

Example Repetition Code.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_1 b_1 b_2 b_2 b_2 b_3 b_3 b_3 b_4 b_4 b_4$.

Error Correction Codes

Error is detected whp. **Do know** where error is.

Example Repetition Code.
To send $b_1 b_2 b_3 b_4$ send $b_1 b_1 b_1 b_2 b_2 b_2 b_3 b_3 b_3 b_4 b_4 b_4$.
Example Alice wants to send 0110. So she sends 000111111000
What could happen?

1. Bob receives 000111111000, of the right form. Bob is confident he got the msg, and he did.
2. Bob receives 000110111000. 2nd triple is 110. Bob corrects to 111 and is confident he got msg. He did.
3. Bob receives 000110111001. 2nd, 4th triple corrected to 111, 000. He is confident he got msg He did.
4. Bob receives 110110111001. 1st triple corrected to 111. He is confident he got the msg. He did not.

“Alice sends” With Generating Matrix

To send b Alice sends (b, b, b). Can express this as:

“Alice sends” With Generating Matrix

To send b Alice sends (b, b, b). Can express this as:
Alice sends b by sending

$$b(1,1,1) = (b, b, b)$$

$(1,1,1)$ is called a **Generating Matrix**. Note that it is 1×3.

“Bob Sees” with Parity Check Matrices

Alice wants to send (b, b, b). There is noise so the msg Bob gets received is $\vec{b} = (b_1, b_2, b_3)$. Bob multiplies by matrix H below.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_1 + b_2 \\ b_1 + b_3 \end{pmatrix}$$

1. If $b_1 = b_2 = b_3$ then $H\vec{b} = (0, 0)$. No errors.
2. If $b_1 \neq b_2 = b_3$ then $H\vec{b} = (1, 1)$. Error in first bit.
3. If $b_2 \neq b_1 = b_3$ then $H\vec{b} = (1, 0)$. Error in second bit.
4. If $b_3 \neq b_2 = b_1$ then $H\vec{b} = (0, 1)$. Error in third bit.

So $H\vec{b}$ tells Bob if there is an error, and if there is, where it is!

Recap and Generalize

$(1, 1, 1)$ is **Generating Matrix G**.

Recap and Generalize

$(1, 1, 1)$ is **Generating Matrix G**.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \text{ is } \textbf{Parity Check Matrix H.}$$

Recap and Generalize

$(1,1,1)$ is **Generating Matrix G**.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$ is **Parity Check Matrix H.**

- Codes 1 bit as 3 bits, so rate is $\frac{1}{3}$.

Recap and Generalize

$(1, 1, 1)$ is **Generating Matrix G**.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$ is **Parity Check Matrix H.**

- Codes 1 bit as 3 bits, so rate is $\frac{1}{3}$.
- Error Correction: Will catch and correct 1 error.

Recap and Generalize

$(1, 1, 1)$ is **Generating Matrix G**.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \text{ is } \textbf{Parity Check Matrix H.}$$

- Codes 1 bit as 3 bits, so rate is $\frac{1}{3}$.
- Error Correction: Will catch and correct 1 error.
- Error Correction: If ≥ 2 errors may not catch them.

Recap and Generalize

$(1,1,1)$ is **Generating Matrix G**.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$ is **Parity Check Matrix H.**

- Codes 1 bit as 3 bits, so rate is $\frac{1}{3}$.
- Error Correction: Will catch and correct 1 error.
- Error Correction: If ≥ 2 errors may not catch them.
- If see bG then can recover b. (Trivial but important for later.)

The (7,4,1) Code Generator Matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ is Generator Matrix } \mathbf{G}.$$

The (7,4,1) Code Generator Matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$ is Generator Matrix $\mathbf{G}$.

Let $\vec{m} = (m_1, m_2, m_3, m_4)$. $\vec{m}G$ is

$$(m_1+m_3+m_4, m_1+m_2+m_3, m_2+m_3+m_4, m_1, m_2, m_3, m_4) = (b_1, b_2, b_3, b_4$$

The (7,4,1) Code Generator Matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$ is Generator Matrix **G**.

Let $\vec{m} = (m_1, m_2, m_3, m_4)$. $\vec{m}G$ is

$$(m_1+m_3+m_4, m_1+m_2+m_3, m_2+m_3+m_4, m_1, m_2, m_3, m_4) = (b_1, b_2, b_3, b_4$$

Note that

$$\begin{aligned} b_1 &= \quad b_4 + b_6 + b_7 \\ b_2 &= \quad b_4 + b_5 + b_6 \\ b_3 &= \quad b_5 + b_6 + b_7 \end{aligned}$$

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

- If all coordinates are 0, then no errors.

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

- If all coordinates are 0, then no errors.
- There are $7 = 2^3 - 1$ ways that $h\vec{b} \neq \vec{0}$. Each one corresponds to which bit is incorrect. (Not obvious.)

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

- If all coordinates are 0, then no errors.
- There are $7 = 2^3 - 1$ ways that $h\vec{b} \neq \vec{0}$. Each one corresponds to which bit is incorrect. (Not obvious.)
- This is an error-correcting code with rate $\frac{4}{7} > \frac{1}{3}$.

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

- If all coordinates are 0, then no errors.
- There are $7 = 2^3 - 1$ ways that $h\vec{b} \neq \vec{0}$. Each one corresponds to which bit is incorrect. (Not obvious.)
- This is an error-correcting code with rate $\frac{4}{7} > \frac{1}{3}$.
- This is a (7,4,1)-code. $|\vec{b}| = 7$, $|\vec{m}| = 4$, corrects 1 error.

The (7,4,1) Code Parity Check Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ is Parity Check Matrix } \mathbf{H}.$$

$\vec{b} = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$. $H\vec{b}$ is
$(b_1 + b_4 + b_6 + b_7, b_2 + b_4 + b_5 + b_6, b_3 + b_5 + b_6 + b_7)$

- If all coordinates are 0, then no errors.
- There are $7 = 2^3 - 1$ ways that $h\vec{b} \neq \vec{0}$. Each one corresponds to which bit is incorrect. (Not obvious.)
- This is an error-correcting code with rate $\frac{4}{7} > \frac{1}{3}$.
- This is a (7,4,1)-code. $|\vec{b}| = 7$, $|\vec{m}| = 4$, corrects 1 error.
- If see $\vec{m}G$ can recover $\vec{m}$ easily: the last four bits.

The (7,4,1) Code or A (7,4,1) Code?

Recall our (7,4,1) Code had a matrix G, and:
If $\vec{m} = (m_1, m_2, m_3, m_4)$ then $\vec{m}G$ is

$$(m_1 + m_3 + m_4, m_1 + m_2 + m_3, m_2 + m_3 + m_4, \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \mathbf{m_4})$$

So the msg is in slots 4,5,6,7 and the error-correction takes place in slots 1,2,3.

The (7,4,1) Code or A (7,4,1) Code?

Recall our (7,4,1) Code had a matrix G, and:
If $\vec{m} = (m_1, m_2, m_3, m_4)$ then $\vec{m}G$ is

$$(m_1 + m_3 + m_4, m_1 + m_2 + m_3, m_2 + m_3 + m_4, \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \mathbf{m_4})$$

So the msg is in slots 4,5,6,7 and the error-correction takes place in slots 1,2,3.

Is there another G such that $\vec{m}G$ is

$$(\mathbf{m_3}, m_1 + m_2 + m_3, \mathbf{m_1}, \mathbf{m_4}, m_1 + m_3 + m_4, m_2 + m_3 + m_4, \mathbf{m_2})$$

Yes.

The (7,4,1) Code or A (7,4,1) Code?

Recall our (7,4,1) Code had a matrix G, and:
If $\vec{m} = (m_1, m_2, m_3, m_4)$ then $\vec{m}G$ is

$$(m_1 + m_3 + m_4, m_1 + m_2 + m_3, m_2 + m_3 + m_4, \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \mathbf{m_4})$$

So the msg is in slots 4,5,6,7 and the error-correction takes place in slots 1,2,3.

Is there another G such that $\vec{m}G$ is

$$(\mathbf{m_3}, m_1 + m_2 + m_3, \mathbf{m_1}, \mathbf{m_4}, m_1 + m_3 + m_4, m_2 + m_3 + m_4, \mathbf{m_2})$$

Yes. Any rearrangement is a (7,4,1) code.

Alice Can Pick a Random (7,4,1) Code

In the protocol we will say:
Alice picks a Random (7,4,1) Code

Alice Can Pick a Random (7,4,1) Code

In the protocol we will say:

Alice picks a Random (7,4,1) Code

This will mean that she randomly permutes the rows of G and the columns of H so that (G, H) is one version of the (7,4,1) code.

If she is given a 7-bit vector with at most one error

Alice Can Pick a Random (7,4,1) Code

In the protocol we will say:

Alice picks a Random (7,4,1) Code

This will mean that she randomly permutes the rows of G and the columns of H so that (G, H) is one version of the (7,4,1) code.

If she is given a 7-bit vector with at most one error

1. Using **H** she can **correct** the vector to the codeword intended.

Alice Can Pick a Random (7,4,1) Code

In the protocol we will say:

Alice picks a Random (7,4,1) Code

This will mean that she randomly permutes the rows of G and the columns of H so that (G, H) is one version of the (7,4,1) code.

If she is given a 7-bit vector with at most one error

1. Using **H** she can **correct** the vector to the codeword intended.
2. Using **G** she can find the msg $\vec{m}$ by using table of what generates what.

Alice Can Pick a Random (7,4,1) Code

In the protocol we will say:

Alice picks a Random (7,4,1) Code

This will mean that she randomly permutes the rows of G and the columns of H so that (G, H) is one version of the (7,4,1) code.

If she is given a 7-bit vector with at most one error

1. Using **H** she can **correct** the vector to the codeword intended.
2. Using **G** she can find the msg $\vec{m}$ by using table of what generates what.

Actually we will use matrices much bigger than (7,4,1).

1-Error Correcting Codes

We assume $n+1$ is a power of 2.

Def An **$(n,k,1)$-Error Correcting Code** is two matrices:

1-Error Correcting Codes

We assume $n+1$ is a power of 2.

Def An **$(n, k, 1)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \to \{0,1\}^n$.

1-Error Correcting Codes

We assume $n+1$ is a power of 2.

Def An **$(n, k, 1)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(n+1)$ is parity check matrix. Output is $\lg_2(n+1)$ bits, so $n+1$ possibilities:

1-Error Correcting Codes

We assume $n+1$ is a power of 2.

Def An **$(n,k,1)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(n+1)$ is parity check matrix. Output is $\lg_2(n+1)$ bits, so $n+1$ possibilities:

   0 errors, 1 possibility.

1-Error Correcting Codes

We assume $n+1$ is a power of 2.

Def An **$(n, k, 1)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(n+1)$ is parity check matrix. Output is $\lg_2(n+1)$ bits, so $n+1$ possibilities:

   0 errors, 1 possibility.

   1-error and where it is, n possibilities.

1-Error Correcting Codes

We assume $n + 1$ is a power of 2.

Def An **$(n, k, 1)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0, 1\}^k \to \{0, 1\}^n$.
2. H is $n \times \lg_2(n + 1)$ is parity check matrix. Output is $\lg_2(n + 1)$ bits, so $n + 1$ possibilities:

   0 errors, 1 possibility.

   1-error and where it is, n possibilities.

What about 2 errors?

2-Error Correcting Codes

We assume $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ is a power of 2.

Def An $(n, k, 2)$**-Error Correcting Code** is two matrices:

2-Error Correcting Codes

We assume $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ is a power of 2.

Def An **$(n, k, 2)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.

2-Error Correcting Codes

We assume $\binom{n}{2}+\binom{n}{1}+\binom{n}{0}$ is a power of 2.
Def An ***(n, k, 2)*-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(\binom{n}{2}+\binom{n}{1}+\binom{n}{0})$ is parity check matrix. Output is $\lg_2(\binom{n}{2}+\binom{n}{1}+\binom{n}{0})$ bits, so $\binom{n}{2}+\binom{n}{1}+\binom{n}{0}$ possibilities:

2-Error Correcting Codes

We assume $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ is a power of 2.

Def An **$(n, k, 2)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0, 1\}^k \rightarrow \{0, 1\}^n$.
2. H is $n \times \lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ is parity check matrix. Output is $\lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ bits, so $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ possibilities:

   0 errors, $\binom{n}{0}$ possibilities.

2-Error Correcting Codes

We assume $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ is a power of 2.

Def An **$(n, k, 2)$-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ is parity check matrix. Output is $\lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ bits, so $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ possibilities:

   0 errors, $\binom{n}{0}$ possibilities.

   1 error and where it is, $\binom{n}{1}$ possibilities.

2-Error Correcting Codes

We assume $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ is a power of 2.

Def An ***(n, k, 2)*-Error Correcting Code** is two matrices:

1. G is $k \times n$. $G : \{0,1\}^k \rightarrow \{0,1\}^n$.
2. H is $n \times \lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ is parity check matrix. Output is $\lg_2(\binom{n}{2} + \binom{n}{1} + \binom{n}{0})$ bits, so $\binom{n}{2} + \binom{n}{1} + \binom{n}{0}$ possibilities:

   0 errors, $\binom{n}{0}$ possibilities.

   1 error and where it is, $\binom{n}{1}$ possibilities.

   2 errors and where they are, $\binom{n}{2}$ possibilities.

t-Error Correcting Codes

I leave the definition of ***t*-Error Correcting Codes** to you.

t-Error Correcting Codes

I leave the definition of ***t*-Error Correcting Codes** to you.

If (G, H) is an error-correcting code then elements in the image of G are **codewords**.

Goppa Codes

Goppa Codes

Goppa Codes

1. They are based on Algebraic Geometry and are very good.

Goppa Codes

Goppa Codes

1. They are based on Algebraic Geometry and are very good.
2. McEliece cipher works with any error correcting code; however, in practice they use Goppa codes.

Goppa Codes

Goppa Codes

1. They are based on Algebraic Geometry and are very good.
2. McEliece cipher works with any error correcting code; however, in practice they use Goppa codes.
3. We will not have to learn Goppa codes to understand McEliece Cipher.

Goppa Codes Parameters

We will present parameters for Goppa Codes.
k is length of msg Alice wants to send
n is length of msg Alice sends.
t is how many errors the code can correct. We want this large.
$R = k/n$ is the rate of the code. We want this large.

Goppa Codes Parameters

We will present parameters for Goppa Codes.
k is length of msg Alice wants to send
n is length of msg Alice sends.
t is how many errors the code can correct. We want this large.
$R = k/n$ is the rate of the code. We want this large.

Here is a table of some known Goppa Code parameters.

n	k	t	$R = k/n$
1024	524	50	0.512
2048	1751	27	0.854
1632	1269	34	0.778

Back to McEliece Public Key Cryptosystem

For the Rest of This Talk

For the rest of this talk:

For the Rest of This Talk

For the rest of this talk:

1. $n, k, t \in \mathbb{N}$ with $k, t < n$.

For the Rest of This Talk

For the rest of this talk:

1. $n, k, t \in \mathbb{N}$ with $k, t < n$.
2. (G, H) is an (n, k, t) Error Corr. code. Alice picks representation of (G, H) at random. k is length of msg Alice wants to send, n is length of what she sends, t is numb of errs corrected.

For the Rest of This Talk

For the rest of this talk:

1. $n, k, t \in \mathbb{N}$ with $k, t < n$.
2. (G, H) is an (n, k, t) Error Corr. code. Alice picks representation of (G, H) at random. k is length of msg Alice wants to send, n is length of what she sends, t is numb of errs corrected.
3. $\vec{e} \in \{0, 1\}^n$ will have weight t, meaning t ones.

For the Rest of This Talk

For the rest of this talk:

1. $n, k, t \in \mathbb{N}$ with $k, t < n$.
2. (G, H) is an (n, k, t) Error Corr. code. Alice picks representation of (G, H) at random. k is length of msg Alice wants to send, n is length of what she sends, t is numb of errs corrected.
3. $\vec{e} \in \{0, 1\}^n$ will have weight t, meaning t ones.
4. The McEliece Public Key Cryptosystem usually uses Goppa Codes. Not important for this talk.

For the Rest of This Talk

For the rest of this talk:

1. $n, k, t \in \mathbb{N}$ with $k, t < n$.
2. (G, H) is an (n, k, t) Error Corr. code. Alice picks representation of (G, H) at random. k is length of msg Alice wants to send, n is length of what she sends, t is numb of errs corrected.
3. $\vec{e} \in \{0, 1\}^n$ will have weight t, meaning t ones.
4. The McEliece Public Key Cryptosystem usually uses Goppa Codes. Not important for this talk.
5. Recall: Everything is mod 2.

An Example of a Perm Matrix

Note that:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} c \\ a \\ b \\ d \end{pmatrix}$$

The matrix **permutes** the input.

Perm Matrices

Def A **Perm Matrix** is a matrix where

1. Every row has one 1.
2. Every column has one 1.
3. Every row is distinct.
4. Every column is distinct (this follows from 1,2,3).

Perm Matrices

Def A **Perm Matrix** is a matrix where

1. Every row has one 1.
2. Every column has one 1.
3. Every row is distinct.
4. Every column is distinct (this follows from 1,2,3).

One can show that

- If P is a perm matrix then P computes a permutation.
- If P computes a permutation then P is a perm matrix.

McEliece Public Key: Alice Preps

McEliece Public Key: Alice Preps

1. Alice picks a random invertible $k \times k$ matrix S.

McEliece Public Key: Alice Preps

1. Alice picks a random invertible $k \times k$ matrix S.
2. Alice picks a random $n \times n$ Perm matrix P.

McEliece Public Key: Alice Preps

1. Alice picks a random invertible $k \times k$ matrix S.
2. Alice picks a random $n \times n$ Perm matrix P.
3. Alice picks at random some (G, H) which works for the code.

McEliece Public Key: Alice Preps

1. Alice picks a random invertible $k \times k$ matrix S.
2. Alice picks a random $n \times n$ Perm matrix P.
3. Alice picks at random some (G, H) which works for the code.
4. **Public** The $k \times n$ matrix SGP.

McEliece Public Key: Alice Preps

1. Alice picks a random invertible $k \times k$ matrix S.
2. Alice picks a random $n \times n$ Perm matrix P.
3. Alice picks at random some (G, H) which works for the code.
4. **Public** The $k \times n$ matrix SGP.
5. **Private** The matrices S and P and the error correcting (n, t, k) code (G, H). (Note: It is known which (n, t, k) code Alice is using, but not which (G, H).)

McEliece Public Key: Bob Sends

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.
5. Alice computes $\vec{y}P^{-1} = \vec{m}SG + \vec{e}P^{-1}$.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.
5. Alice computes $\vec{y}P^{-1} = \vec{m}SG + \vec{e}P^{-1}$.
6. $\vec{m}SG$ is a codeword. $\vec{e}P^{-1}$ has weight t.
   So $\vec{m}SG + \vec{e}P^{-1}$ is t away from a codeword.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.
5. Alice computes $\vec{y}P^{-1} = \vec{m}SG + \vec{e}P^{-1}$.
6. $\vec{m}SG$ is a codeword. $\vec{e}P^{-1}$ has weight t.
   So $\vec{m}SG + \vec{e}P^{-1}$ is t away from a codeword.
7. Alice has **H** so can recover the codeword $\vec{m}SG$.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.
5. Alice computes $\vec{y}P^{-1} = \vec{m}SG + \vec{e}P^{-1}$.
6. $\vec{m}SG$ is a codeword. $\vec{e}P^{-1}$ has weight t.
   So $\vec{m}SG + \vec{e}P^{-1}$ is t away from a codeword.
7. Alice has **H** so can recover the codeword $\vec{m}SG$.
8. Alice has **G** so can recover $\vec{m}S$. She knows which bits are $\vec{m}$.

McEliece Public Key: Bob Sends

1. Bob (and Eve) have SGP which is a $k \times n$ matrix.
2. Bob wants to send k bits $\vec{m} = m_1 \cdots m_k$.
3. Bob computes $\vec{m}SGP$ and random $\vec{e}$ of weight t.
4. Bob sends $\vec{y} = \vec{m}SGP + \vec{e}$. Note that this is t-away from $\vec{m}SGP$, but $\vec{m}SGP$ is not necc. a codeword.
5. Alice computes $\vec{y}P^{-1} = \vec{m}SG + \vec{e}P^{-1}$.
6. $\vec{m}SG$ is a codeword. $\vec{e}P^{-1}$ has weight t.
   So $\vec{m}SG + \vec{e}P^{-1}$ is t away from a codeword.
7. Alice has **H** so can recover the codeword $\vec{m}SG$.
8. Alice has **G** so can recover $\vec{m}S$. She knows which bits are $\vec{m}$.
9. Multiply by S^{-1} to get $\vec{m}$.

Secure?

Eve has SGP.

Secure?

Eve has SGP.

1. There are many matrices whose product is the same as SGP.

Secure?

Eve has SGP.

1. There are many matrices whose product is the same as SGP.
2. Believed to be hard to find S, G, P.

Secure?

Eve has SGP.

1. There are many matrices whose product is the same as SGP.
2. Believed to be hard to find S, G, P.

Eve has $\vec{m}SGP + \vec{e}$.

Secure?

Eve has SGP.

1. There are many matrices whose product is the same as SGP.
2. Believed to be hard to find S, G, P.

Eve has $\vec{m}SGP + \vec{e}$.

1. Hard to error correct without H. This is real point.

Secure?

Eve has SGP.

1. There are many matrices whose product is the same as SGP.
2. Believed to be hard to find S, G, P.

Eve has $\vec{m}SGP + \vec{e}$.

1. Hard to error correct without H. This is real point.
2. Hard to find $\vec{m}$ without P and G.

PROS and CONS

PRO Does not rely on factoring or discrete log or any other problem in Number Theory being hard. Why is this good?

PROS and CONS

PRO Does not rely on factoring or discrete log or any other problem in Number Theory being hard. Why is this good?

PROS and CONS

PRO Does not rely on factoring or discrete log or any other problem in Number Theory being hard. Why is this good?

1. The usual reason given: Factoring might end up being easy via **Quantum** or **Erika-Guido-Natalyia**.

PROS and CONS

PRO Does not rely on factoring or discrete log or any other problem in Number Theory being hard. Why is this good?

1. The usual reason given: Factoring might end up being easy via **Quantum** or **Erika-Guido-Natalyia**.
2. We may find that McEliece or LWE might have properties that make it **better** thatn RSA. This will require using it for a while.

CON Since McEliece and LWE have not been out there much they have not been truly tested.
CON RSA, even with the stuff you do to make it really work, seems easier to code up then McEliece. For LWE its harder to say.

PROS and CONS

PRO Does not rely on factoring or discrete log or any other problem in Number Theory being hard. Why is this good?

1. The usual reason given: Factoring might end up being easy via **Quantum** or **Erika-Guido-Natalyia**.
2. We may find that McEliece or LWE might have properties that make it **better** thatn RSA. This will require using it for a while.

CON Since McEliece and LWE have not been out there much they have not been truly tested.
CON RSA, even with the stuff you do to make it really work, seems easier to code up then McEliece. For LWE its harder to say. Especially if the NSA is listening in.

My Real World Security Issues

Email I got recently

Attention: Owner of the Fund, We are delegates of the IMF in conjunction with the assistance of the UN of the AU, the EU and the FBI to pay victims of fraud 3.7 million dollars each. In the course of our investigation, The UN Commission against Crime and the IMF ordered that the money recovered from the scammers be distributed among 10 lucky people around the world. World for compensation. **This email / letter has been sent to you because your email address was found in one of the scam artists' files and the computer is hard drive during our investigation, maybe you were scammed or not, it is being compensated with the sum of us $3,700,000.** Reconfirm your information as indicated below. 1,Full Names name 2,Contact Address, 3. Nationality, 4. State of origin. Mr Victor Markson

Article I Read Recently

Detecting Phishing Attempts

dl.acm.org/doi/10.1145/3415231

Abstract To better understand the cognitive process that end users can use to identify phishing msgs, I interviewed 21 IT experts about instances where they successfully identified emails as phishing in their own inboxes. IT experts naturally follow a three-stage process for identifying phishing emails. (1) the email recipient tries to make sense of the email (2) they notice discrepancies: little things that are **off about the email** (3) some feature of the email – usually, the presence of a link requesting an action – triggers them to recognize that phishing is a possible alternative explanation.

Article I Read Recently

Detecting Phishing Attempts

`dl.acm.org/doi/10.1145/3415231`

Abstract To better understand the cognitive process that end users can use to identify phishing msgs, I interviewed 21 IT experts about instances where they successfully identified emails as phishing in their own inboxes. IT experts naturally follow a three-stage process for identifying phishing emails. (1) the email recipient tries to make sense of the email (2) they notice discrepancies: little things that are **off about the email** (3) some feature of the email – usually, the presence of a link requesting an action – triggers them to recognize that phishing is a possible alternative explanation.

off about the email Offering me $3,700,000 seemed just a little bit off.

Another Email I Got (excepts)

Urgent - help me distribute my $12 million to humanitarian aid.

This mail might come to you as a surprise and **the temptation to ignore it as unserious could come into your mind but please consider it a divine wish and accept it with a deep sense of humility.**

Since the loss of my husband and also because i had no child to call my own, i have found a new desire to assist the helpless. **I have donated some money to orphans** in Sudan, Ethiopia, Cameroon, Spain, Austria, Germany and some Asian countries.

I have **12,000,000.00 u. S. Dollars** which i deposited in a security company in Cotonou Benin Republic that **does not know the real content** to be money and i want you to assist me in claiming the consignment & distributing the money to charity organizations, **i agree to reward you with part of the money** for your assistance, kindness and participation in this godly project.

i am in the hospital where i have been undergoing treatment for **oesophageal cancer** and my doctors have told me that i have only a few months to live.

Why is Spam Harmful?

Why is Spam Harmful?

1. Might fool some people.

Why is Spam Harmful?

1. Might fool some people.
2. Wastes the time of all

Why is Spam Harmful?

1. Might fool some people.
2. Wastes the time of all
3. It makes it hard to tell who is legit. If I get a letter from a charity I tend to throw it away assuming it is spam.

Why is Spam Harmful?

1. Might fool some people.
2. Wastes the time of all
3. It makes it hard to tell who is legit. If I get a letter from a charity I tend to throw it away assuming it is spam.
4. I can't tell the **real Nigerian billionairs** who want to give me $12,000,000 from the **fake ones**!

STOP RECORDING LECTURE