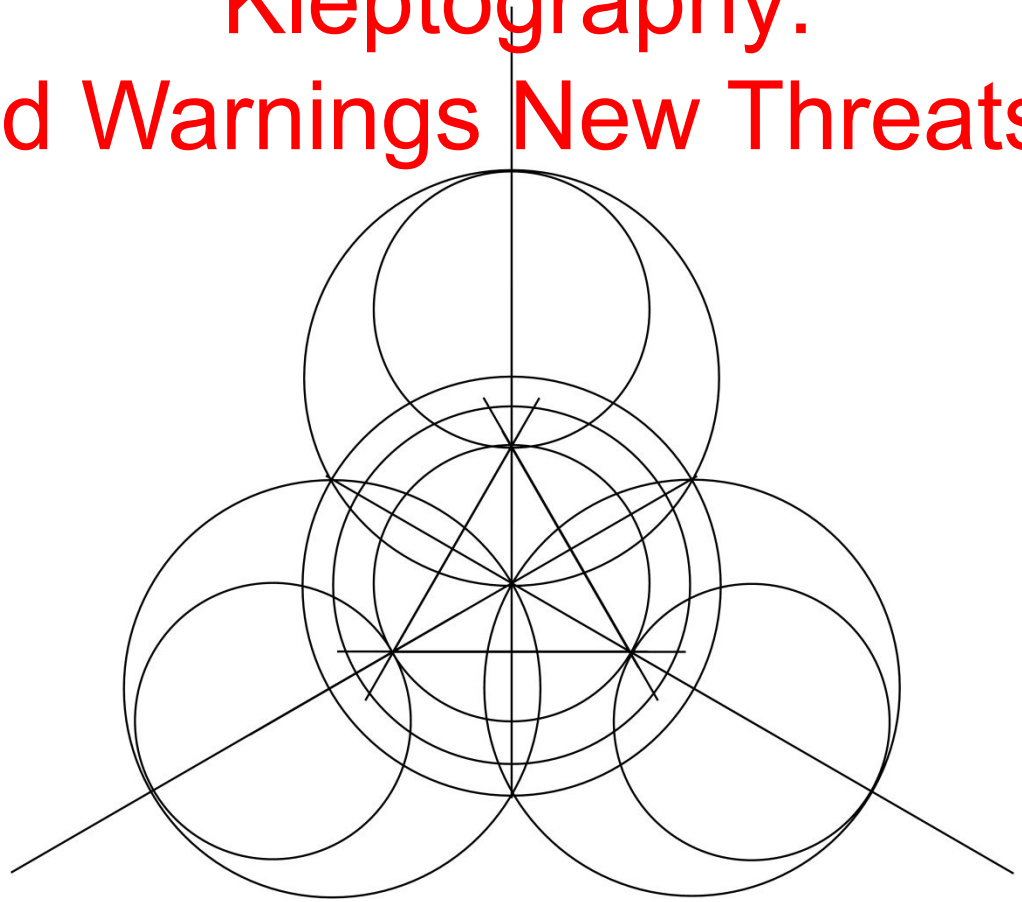
Kleptography:
Old Warnings New Threats!!



MOTI YUNG

Google Inc. / Columbia University

Joint work with **Adam Young**

Background:

-The time is the Mid 90's: Cryptography is the big Equalizer (the small guys against governments); CRYPTO-WAR was won (no capstone, no escrow!, Clipper chip attacked); popular books on crypto (almost) make everyone (believe she/he is) a cryptographer.

-For 1000's of years Crypto was used as a protective technology, and still is!

We started to investigate use of crypto as an attack technology!! OLD WARNINGS...

(resulting in cryptovirology and kleptography)

There are simple attacks..

- **Fix the Pseudorandom Generator (inside the crypto box)**
- **Use weak algorithm version (inside the box)**

Indeed, in many implementations these issues are not even checked against, but they lead to "easy reverse engineering" and are "detectable" from the I/O relations (as we saw recently in the field).....

We thought this "algorithm replacement attack" or symmetric backdoor attacks are easy (and relatively understood)..

Note: employing these– will break security globally, i.e., will weaken the Internet....

How hard/ easy are undetectable (exclusive) backdoors ????

Public key is Great: RSA Encryption/Decryption

- N=p*q, where p,q are large primes known to the key owner
- Everyone knows N and e.
- Let d be the private exponent where ed = 1 mod (p-1)(q-1)
- To encrypt $m \in Z_n^*$ (after padding..) compute: $c = m^e \bmod n$
- To decrypt the ciphertext c compute: $m = c^d \bmod n$
- As far as we know: Only with known factorization given N and e, one can find d.
- Asymmetry: only the key owner can decrypt
- Asymmetry is great: paradoxical protocols like Oblivious Transfer, Mental poker, etc. can be implemented

Kleptography Motivation: Crypto on the Attack

- **Can we have cryptographic Trojan horses (backdoors) that are robust against reverse-engineering. They are only useful to the attacker (asymmetric backdoors!) when that attacker is an eavesdropper (as in surveillance!)**
- **This will keep global security while providing the attacker with exclusive use of insecurity**

- **First Paper noted: Certain Organizations Will Love It!**
- **Some criticism initially: well, there is enough easy attacks, so why is this... This is theory.....** ☹
- **But, some people understood the motivation!** ☺

What is Kleptography?

- Kleptography is the study of stealing information securely (exclusively), efficiently and subliminally (unnoticeably).
- Stealing from your most trusted “hardware protected systems,” “un-scrutinized software,” (wherever you do not access the algorithm) etc. [Stealing your keys/ secrets]
- It employs Crypto against Crypto! Hiding Crypto in Crypto (as steganography text hiding text)....

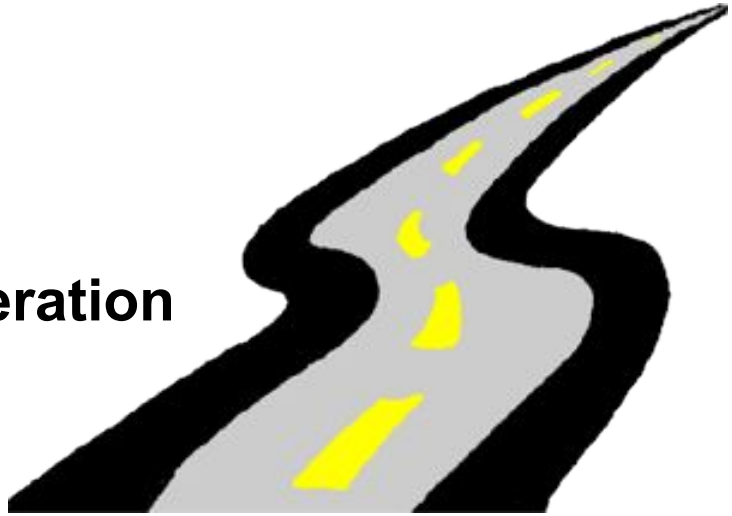
  i.e., Exploiting the asymmetry in PKC

-Note: Both Kleptography and Timing-Attacks (physical leakage) presented first at Crypto’96 !

The goal

- To develop a **robust** backdoor within a cryptosystem that:
  1) **EXCLUSIVITY: Provides only to the attacker the desired secret information (e.g., private key of the unwary user)**
  2) **INDISTINGUISHABILITY: Cannot be detected in black-box implementations (I/O access only as in tamper-resistant systems) except by the attacker**
  3) **FORWARD SECRECY: If a reverse-engineer (i.e., not the attacker) breaches the black-box, then the previously stolen information remains confidential (secure against reverse-engineering). Ideally, confidentiality holds going forward as well (backword security) if the exposure is temporary.**
- **The successful reverse-engineer** may learn that the attack is carried out (different algorithm is run), BUT will be **unable** to use the backdoor.
- Note: asymmetry between “attacker” and “reverse engineer/ others”

Talk Road Map



Warnings:

- Kleptographic attack on **RSA key generation**
- Definition of a Secretly Embedded Trapdoor with Universal Protection (**SETUP**)
- Kleptographic attack on the **Diffie-Hellman** key exchange
- Implications to trust relationships
- Recent Threats....

Kleptographic Theft of RSA Private Key

- **Problem: To devise a backdoor (i.e., a way to covertly obtain the RSA private keys of users) that can be deployed in an RSA [RSA78] key generation program such that:**
  - **The resulting RSA key pair must "look like" a normal RSA key pair (indistinguishability).**
  - **The same copy of the key generation program is obtained by everyone (it may be code signed for instance).**
- **Note that a pseudorandom bit generator that uses a fixed secret seed does not accomplish this. The seed or seeds will be revealed to the reverse-engineer and the resulting pseudorandom bit sequences will be revealed.**
- **Applies t all factoring-based systems in any context!**

Normal RSA Key Generation

- **Let e be the public RSA exponent that is shared by all the users (e.g., e is often taken to be $2^{16}+1$ or 3)**

1) **choose a large number p randomly (e.g., p is 1024 bits long)**
2) **if p is composite or gcd(e,p - 1) ≠ 1 then goto step 1**
3) **choose a large number q randomly**
4) **if q is composite or gcd(e,q - 1) ≠ 1 then goto step 3**
5) **output the public key (n=pq,e) and the private key p**

Note that the private exponent d is found by solving for (d,k) in ed + kφ(n) = 1 (using the *extended Euclidean alg*.)

RSA Keys we will use in the replaced Alg.

- There will be a owner public key N=p*q (say of size 2048 bits).
- There will be an RSA key of the attacker which we will call Y which will be of half the size (say, 1024 bits),Y=y1 * y2, where y1, y2 are large prime numbers of size 512 bits.

Kleptographic RSA Key Generation

- **The key generation algorithm is modified to contain a cryptotrojan. The cryptotrojan contains *** the attacker's RSA public key Y. This is an earlier version of the attack [YY96,YY97], more mature versions exist [YY04,YY05].**

1) **choose a large value s randomly (e.g., 1024-bits)**
2) **compute p = H(s) where H is a cryptographic one-way function**
3) **if p is composite then goto step 1**
4) **choose a large 1024-bit string RND randomly**
5) **compute c to be an encryption of s under RSA public Y (1024 bit RSA key- Y half the size of n) c= s^e mod Y**
6) **solve for (q,r) in (c || RND) = pq + r {int-div: q quotient, r remainder }**
7) **if q is composite then goto step 1**
8) **output the public key (n=pq,e) and the private key p**

Note that n is ~2048 bits in length

c = Encryption by attacker's RSA key Y of half the size of n of the plaintext s

(c || RND) = pq + r → (c || RND) – r = pq = n

Note that r is about sqrt of n thus the (– r) operation will not ruin c by more than one bit (the borrow bit).

→ The value c is not hidden much by the high order bits of n

The fact that p and q so chosen are likely to be primes is by the prime number theorem.

Recovering the RSA Private Key

- **The private key is recovered as follows:**
  - **The attacker obtains the public key (n,e) of the user**
  - **Let u be the 1024 uppermost bits of n**
  - **The attacker sets c_1 = u and c_2 = u+1 (c_2 accounts for a potential borrow bit having been taken from the computation $n = pq = (c \,||\, RND) - r$**
  - **The attacker (knowing the private key of Y) decrypts c_1 and c_2 to get s_1 and s_2, respectively (**)**
  - **Either $p_1 = H(s_1)$ or $p_2 = H(s_2)$ will divide n evenly**
- **Only the attacker can perform this operation since only the attacker knows the needed private decryption key in (**).**

Definition of a SETUP

A SETUP attack is an algorithmic modification C’ of a cryptosystem C with the following properties:

1) **Halting Correctness: C and C' are efficient algorithms.**
2) **Output Indistinguishability: The outputs of C and C' are computationally indistinguishable to all efficient algorithms except for the attacker A.**
3) **Confidentiality of C: The outputs of C do not compromise the security of the cryptosystem that C implements.**
4) **Confidentiality of C': The outputs of C' only compromise the security of the cryptosystem that C’ implements with respect to the attacker A.**
5) **Ability to compromise C': With overwhelming probability the attacker A can break/ decrypt/ cryptanalyze at least one private output of C' given a sufficient number of public outputs of C'.**

Formal Aspects

- **There is a formal "security model and definitions"**
- **The design employs tools of modern cryptography: indistinguishability, careful probability distributions, pseudorandomness and random oracle assumptions, etc.**
- **There is a proof of security of the design (in the model). The proof is more complicated than in regular systems (we have two systems in one):**
  - **RSA is a good key (the regular proof p,q are random primes)**
  - **The hidden channel is secure (subliminal and exclusive)**

It is "fun" to use formal methodology and techniques to prove the "security of klepto," almost having <u>"provable insecurity"</u> ☺ ☺

But..

- The security for the attacker is of half the size key of that of the user…Y is half the size of n…
- Can we do anything? … we will see… this was only the first work..

Diffie-Hellman Key Exchange Parameters

- Applies generically as a general scheme.

- Concrete parameters (example): Let p be a large prime
- Let g < p be an element in Z_p^* with order q
- (p,q) must provide a suitable setting for the *discrete-logarithm problem (a typical setting is p=2q+1, p,q primes but also smaller prime order subgroups possible)*.
- The parameters (p,q) are public

The Diffie-Hellman Key Exchange

1) Alice chooses $a < q$ randomly
2) Alice sends $A = g^{a} \bmod p$ to Bob

3) Bob chooses $b < q$ randomly
4) Bob sends $B = g^{b} \bmod p$ to Alice

5) Alice computes $k = B^{a} \bmod p$
6) Bob computes $k = A^{b} \bmod p$

Observe that $k = B^{a} = A^{b} \bmod p$ since $g^{ba} = g^{ab} \bmod p$

The Diffie-Hellman Assumption

- The classic Diffie-Hellman key exchange relies on the presumed intractability, under the DDH (in certain groups) from g^a and g^b entire g^{a*b} is random
- Asymmetry: if you know a or b, you get the random value from the exchange !!!!

The RSA key generation has a large subliminal channel (half of the bits can be fixed and we get a composite N) as was noted earlier [Desmedt, A. Lenstra,..]. The DH problem does not have subliminal channel that is large enough [as noted by Simmons] (under the decisional assumption all bits are equally random and useful)…

Do we need subliminal channel? Is DDH making DH secure wrt setup?

So.. Is subliminal channel needed?

- The computer Science Answer: If there isn't one create one!
- It's the science of the artificial/ engineering, after all…….

The setup channel is a channel between the device and the attacker and there are other ways to establish "secure communication channels" when crypto is involved….

Setting for the DH SETUP attack

- The setting is as follows:

1) The attacker can deploy the SETUP attack in Alice device.
2) The malicious designer can act as a passive eavesdropper on all of Alice and Bob's key exchanges.
3) The black-box can store state information across invocations of the Diffie-Hellman algorithm (non-volatile memory).

Goal of the SETUP attack against DH

- The goals of the **simplified** SETUP attack are:

1) To permit the malicious manufacturer to learn every other (or all but one) Diffie-Hellman shared secret k that Alice and Bob compute.

2) To prevent Alice and Bob (and everyone else) from knowing that the attack is taking place without reverse eng.

3) Robustness against reverse-engineering:
   - **If only the code for the SETUP attack is disclosed then all shared secrets past and future will remain confidential.**
   - **A single DH shared secret may be compromised if the non-volatile state information is disclosed.**

Parameters for the DH SETUP attack

- Parameters for the attack:

x_m: private key generated by the malicious attacker for the attack. x_m is randomly chosen such that $x_m < q$ and x_m is kept secret by the attacker (e.g., in the attacker's smart card).

y_m: public key corresponding to x_m. Hence, $y_m = g^{x_m} \bmod p$. y_m is placed inside the black-box that Alice uses.

H: public cryptographic one-way hash function such that:

$$H: \{0,1\}^* \rightarrow Z_q$$

Intuition behind the DH SETUP attack

The idea is to have the attacker:

1) Generate a private key x_m and public key $y_m = g^{x_m} \bmod p$
2) Place the public key y_m in the black-box
3) Design the black-box to compute a shared secret k between Alice and the attacker in the 1st DH key exchange between Alice and Bob where a is the secret exponent, i.e., compute.

$$\mathbf{k = y_m^{\,a} \bmod p}$$

- Then use pseudorandomness derived from k **instead of** a random exponent in Alice's 2d key exchange.

This allows the attacker to learn the second Diffie-Hellman shared secret!

The Diffie-Hellman SETUP Attack

First exchange:----------

- **Alice's device sends $A_1 = g^{a_1} \bmod p$ to Bob where $a_1 \in_R Z_q$**
- **Alice's device stores a_1 in non-volatile memory**
- **Bob's device sends $B_1 = g^{b_1} \bmod p$ to Alice where $b_1 \in_R Z_q$**
- **Alice and Bob's devices compute $k_1 = g^{a_1 b_1} \bmod p$**

Second exchange:----------

- **Alice's device computes $a_2 = H(y_m{}^{a_1} \bmod p)$- pseudorandom**
- **Alice's device sends $A_2 = g^{a_2} \bmod p$ to Bob**
- **Bob's device sends $B_2 = g^{b_2} \bmod p$ to Alice where $b_2 \in_R Z_q$**
- **Alice and Bob's devices compute $k_2 = g^{a_2 b_2} \bmod p$**

Recovering the 2nd DH Shared Secret

The attacker:

1) Obtains A_1 and B_2 via first passive eavesdropping.
2) Computes $a_2 = H(A_1^{x_m} \bmod p)$
3) Computes $k_2 = B_2^{a_2} \bmod p$

Note that attack uses the very commutativity of the DH problem:

The attacker computes $A_1^{x_m} \bmod p = g^{a_1 x_m} = g^{x_m a_1}$

$= y_m^{a_1} \bmod p$ as Alice computes

In two DH key exchanges three exchanges are run !!!!

Security of the DH SETUP attack

- **Confidentiality w.r.t. the reverse-engineer:**
  - **The reverse-engineer learns y_m (we may assume that a_1 is learned and so at most a_2 is compromised, if not yet erased, so erasure is important).**
  - **The reverse-engineer still must solve instances of the Diffie-Hellman problem to learn past DH shared secrets k_2.**

Repeated SETUP: Chaining the DH attack

- The attack generalizes to reveal t out of t+1 Diffie-Hellman shared secrets (larger window of exposure to reverse-engineer).
- This is accomplished by chaining the use of the DH pseudorandom exponent.
- For example:

$$a_3 = H(y_m^{a_2} \bmod p)$$

$$a_4 = H(y_m^{a_3} \bmod p)$$

.....

- This is called a (t,t+1)-SETUP attack (or: repeated setup)

Attack Variation applied to

- Many of the DH based encryption and signature systems: ElGamal, CS, etc. ElGamal sig.,
- DSA signatures: even if the signature sizes are 320 bits (two elements of small subgroup), can extract the secret key of large group size (say 1024 bits) from two signatures… since in the middle of verification we have a full size… g^r
- Other Algebraic structures (e.g., elliptic curves, with curve multiplication as the “exponentiation” operation)…

Then: Small Space Kleptogram in RSA KeyGen: Intuition Behind the Approach

- Elliptic Curve Cryptography gives smaller ciphertexts (with point compression) than RSA with a comparable security parameter. This helps RSA key generation where the security of attacker matches that of the key! (since small EC's have larger security than factoring).
- The use of a **twisted pair of binary curves** gives a Diffie-Hellman key exchange value that is (essentially) a bit string selected uniformly at random, for hiding the key exchange bits in uniorm string (crucial to drawing random RSA instances). Twisted-Dual-EC suggested for use...Other constructions are possible...

  → This suggests that we can embed a DH key exchange value in the upper order bits of n = pq and achieve indistinguishability of RSA backdoor public keys vs. "normal" public keys.

More Recently

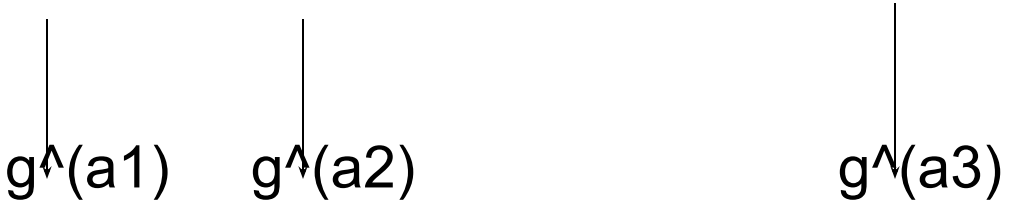
- All other attacks needed hiding of one cryptographic tool inside another. To argue that the attack “looks like” the regular version, Random Oracle like arguments were used (ideal distribution).
- Can we achieve attack on RSA keys that is a property of cryptographic tools only without idealization to a random oracle? (this gives a pure algebraic statement on strange relations among two cryptographic distributions, one living inside another!)
- Recently: Yes…. (in small space i.e., high security against reverse engineering)…employing a novel “random key exchange” again embedding strings derived from EC group as high order strings inside the representation of an RSA composite

New Threats

- What has changed recently?

The (t,t+1) DH (Repeated) Kleptogram

- given: a1 $\rightarrow a_2 = H(y_m^{a1} \bmod p) \rightarrow a_3 = H(y_m^{a_2} \bmod p) \rightarrow . \ldots$



- This is the “repeated DH Setup”
- Top line interpretation: advancing state of prng
- Bottom line: output of prng (used in the exchange in DH setup).

Backdoor: attacker derives top position i+1 from bottom position i

This is the logic behind the Dual-EC prng aleged backdoor.
Recovery w/ “randomly generated points” validated recently !!!!

Sketch: Dual-EC generator

- given: a1=s $\rightarrow a_2 =$ a1* Q $\rightarrow a_3 =$ a2*Q $\rightarrow$. ... (internal)

- Some output bits cut (to make the result random) at each stage
- This is the “repeated DH Setup” if we know Q=s*P
- Top line interpretation: advancing state of prng
- Bottom line: output of prng (rather than DH value)

Aleged Backdoor: attacker derives top position i+1 from bottom position i

Recently: reserchers did full investigation with P and Q chosen related way, and effect on SSL/TLS

What does this mean: Mind Obfuscation!!

- The attacker has obvious plausible deniability: if P and Q are random proof of security of the prng exists! This is the nature of klepto !!!!!!!
- The attacker preentered the algorithm as a new methods for randomness, covertly from random i block gets internal state i+1 and all randomness→ with SSL/TLS random nonce the disclosure of state is built-in!!!
- The attacker did not have to be too creative to make the algorithm, just to algebraically obfuscate it (Crypto97) a bit
- Proof of security implied it looked heavy but secure totally ignoring the fact that kleptographic attack has proofs against all but the attacker!

Cliptography: Clipping the power of Kleptography

In recent years:

-We use some

* Trust assumptions (minimal) in the architecture

* Watchdog element

To design Cryptosystems that prove they follow their specification (so, at least, if spec is checked against subversion, so is the implementation, or correct subverted implementation.

-In general architectures that exhibits faithfulness to the spec's are desired!

Conclusion- technical summary

- The notion of a cryptographic backdoor (subversion) that is robust against reverse-engineering was introduced (SETUP). Assymetry between attacker and others can be find. Channels to convey attack values can be found as well....
  - SETUP attacks against RSA key generation was presented.
  - SETUP attacks against Diffie-Hellman was presented. It applies to many systems… (DSA,…etc.).
  - Application of the DH (ECC) to RSA setup.
- Hiding the algorithm in hw (or its spec's or its constants, or...)...which may have some advantages (not informing attackers)! … also has a "dark side" when combined with surveillance.
- Presenting an algorithm with hidden random constants similarly dangerous !!!!!!!

Conclusions– cryptographic systems

- In all these schemes: we have proof of security of the system (against all but the attacker) and a second security (exclusivity) proof for the attacker [Two systems in one!] and proofs are according to modern standards….
- Cryptography is about security (we know..), it is about solving seemingly paradoxical schemes (we know….), and is also about looking for things that no one will ever look at (thus it is also about non-trivial scrutiny, namely: “hacking with purpose at strange places”). Always be on alert!!
- Attacks on cryptosystems may come from different directions (implementation, hidden malicious parties, physical leakage…. And MALICIOUS DESIGNER/ IMPLEMENTOR). Cryptographic thinking should apply to all layers/ stages.
- Attacks evolve → Crypto must evolve!!!!!!

Conclusions– About Trust

- Trust relationships: manufacturer has to be trusted (not merely the fact that it is a tamper-resistant design that works ok/ tested) …and implementations scrutinized as much as possible (also of software).
- Scrutiny of systems must employ klepto-thinking... “where is the catch”?
- Trust within and about cryptographic system is “tricky” (also true in dealing with other systems, but not everyone thinks about it seriously! So cryptographer ought to look at these other things like trusted platforms…). Know your algorithms (control & values)...
- Black-Box Testing cryptography and trusting it – is it possible??? What about more general security systems? Attacker can only attack after the system is working for a while or just one out of 1000 times (in financial case it may be worth it).
  - Negative results on testing, forensics, etc were derived...
- Beware! … the dual use nature of technology……and always “Expect the Unexpected!”

THANK YOU!