Secure Wireless Communication with Dynamic Secrets

†Sheng Xiao, †Weibo Gong and ‡Don Towsley
†Department of Electrical and Computer Engineering
‡Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01002

***Abstract*—This paper introduces a set of low-complexity algorithms that when coupled with link layer retransmission mechanisms, strengthen wireless communication security. Our basic idea is to generate a series of secrets from inevitable transmission errors and other random factors in wireless communications. Because these secrets are constantly extracted from the communication process in realtime, we call them *dynamic secrets*.**

Dynamic secrets have interesting security properties. They offer a complementary mechanism to existing security protocols. Even if the adversary exploits a vulnerability and steals the underlying system secret, security can be automatically replenished. In many scenarios, it is also possible to bootstrap a secure communication with the dynamic secrets.

I. Introduction

One of the ultimate goals for adversaries to compromise a secure wireless communication system is revealing its underlying system secrets, such as the secret symmetric key or the private key in a public key infrastructure. Once the underlying secret is known by the adversary, communication security will be imperiled. While the system secret is safe, the wireless communication is secured by many research results. However, we are less confident about the communication security when the underlying secrets could be stolen.

Wireless communications contain many potential vulnerabilities that can be exploited to steal the secrets. There is a large variety of wireless devices and they are often mobile. The average user often makes operational mistakes. System secrets leak out due to the complexity of existing security protocols. Wireless adversaries have almost risk-free access to the broadcast signals from and to the target wireless device. They can launch highly aggressive attacks at very low cost. It is extremely difficult to envisage a flawless security setting that can hold a secret forever. Also a wireless user often is not aware that the key is stolen. Therefore, a security mechanism that can survive in the case of possible secret leakages is highly desirable.

When it is possible to steal the underlying secret, frequent session key exchanges do not help much. No matter whether the key exchange uses symmetric or asymmetric encryption, once the underlying secret is stolen, the subsequently session key exchanges become meaningless.

A feasible solution is to frequently exchange the secret using public key cryptography, each time using a new randomly generated public-private key pair. However, the flexibility and versatility of wireless communications restrict the applicability of this solution. Many wireless devices lack the computing power needed for prime number generation. Many scenarios cannot tolerate the delays associated with for public key operations. The complexity of public key infrastructure also makes it less favorable as a general, easy-to-adopt solution for wireless devices.

Our research proposes an inherent, light-weight solution that can protect system secrets and automatically replenish security after secret leakage in wireless communication. It requires little computing power and only relies on the simple fact that wireless communications is error-pone. Hence this solution is widely applicable.

We propose to generate a series of hash values, namely the *dynamic secrets*, based on the link layer communications between wireless devices. Once a dynamic secret is generated, the system secret is updated by XOR with a newly generated dynamic secret. When the adversary suffers from wireless transmission errors and lose information about the dynamic secrets, it will lose information about the system secret too.

Using the adversary's information loss to protect the system secret has two advantages. First, information loss is not recoverable by any computational effort. Second and more important, information loss can be accumulated. Even if the adversary could intrude and seize the system secret at some points of time, after a short additional period of time, the accumulated information loss would void his knowledge about the constantly changing system secrets.

From another point of view, the usage of dynamic secrets changes the attack-defend model in wireless communications. In a conventional setting, one operational mistake or a single vulnerability is sufficient to dismantle the entire security. This is the so called *single point of failure* problem. The adversary predominates in this situation because it can choose any weakness to attack while the user needs to defend against all possibilities. With the help of dynamic secrets, the adversary must fight against any factor that may cause information loss. It is the adversary that suffers a single point of failure.

Consider a scenario where an adversary successfully steals the system secret at time t_o. Because subsequent communications keeps producing dynamic secrets and updating the system secret, the adversary must monitor every bit of information

transmitted for all $t > t_o$. Even if the adversary can eavesdrop on the communication all the time, any receiving error will ruin his effort and prevent him from tracking the secrets.

Forcing the attacker into the single point of failure problem introduces interesting security properties. For example, dynamic secrets could be used to bootstrap the secure communication between wireless nodes in applicable scenarios. We will elaborate these properties in later sections.

The rest of the paper is organized as follows: Section II reviews previous work that motivates our idea and marks our contributions; Section III proposes a simple model to demonstrate how dynamic secrets improve communication security; Section IV introduces algorithms to extract dynamic secrets from link layer wireless communication and analyzes the dynamic secrets from an information theoretical perspective; Section V discusses how to utilize the dynamic secrets to protect the system secrets; Section VI extends the usage of dynamic secrets to help bootstrap security in applicable scenarios; Section VII illustrates our prototype implementation in a wireless LAN; Section VIII summarizes and concludes the paper.

II. Related Works

Our idea originates from the information theoretical research, especially the research on using wireless physical channel properties for secret sharing.

There has been considerable work on the wireless secret sharing problem. The idea of using wireless physical channel properties is intuitive. If the the adversary must have precise knowledge about the channel between legitimate users to determine the shared secrets, then any channel property that is unpredictable to the adversary can be used for secret sharing. [1][2][9][14][24] propose the use of channel reciprocity to generate secret key for wireless communications. The reciprocity between legitimate users cannot be deduced by adversary's observation from his location. Because fading is assumed to be independent for different channels, [3] suggests exploiting channel fades for secret sharing. When the adversary is in deep fading and legitimate users are not, the information exchanged between legitimate users at this moment is privileged and can be manipulated into a shared secret. Under the assumption that legitimate users have perfect channel knowledge, [5][6] propose to use capacity achieving codes for the secret sharing. [26] suggests Detectable Non-Correctable (DNC) codes for the same objective. The channel knowledge assumption is relaxed at the cost of secret sharing efficiency. [22][25] combine channel coding with reliable communication mechanisms to share secrets. [12][13] use antenna diversity to disseminate secrets. These approaches yield unconditional security for the shared secrets. However, they usually demand special hardware upgrades or at least specific interfaces to provide channel measurement information. We cannot directly apply them to protect the system secret for many wireless applications.

The above physical layer approaches can be unified in a framework which had been investigated by Maurer [15][16][17][18]. In the classical Alice, Bob, and Eve scenario, jointly distributed random sources X_A, X_B, and X_E represent Alice, Bob, and Eve's knowledge in the secret sharing respectively. Alice and Bob's goal is to extract a common secret S that Eve does not know, through a publicly known extraction process. There are two steps to achieve this goal. The first step is to find Alice and Bob's common knowledge X_{AB} which could be partially known to Eve. This step is called *reconciliation*. Then Alice and Bob compress X_{AB} into a shorter secret S which is almost completely unknown to Eve. This step is *privacy amplification*. It usually involves universal hashing [7] or random extractor [8][23].

This framework shows that if an adversary can lose information due to randomness in the channel, this randomness can be used for secrecy sharing. We apply this framework to the link layer to generate dynamic secrets. The error retransmission mechanism automatically reconciles the information communicated between sender and receiver. Our algorithms select the subset of information which is received correctly without retransmission and then apply privacy amplification on this subset to generate a dynamic secret. As communication goes on, repeating these algorithms will produce a series of dynamic secrets. In wireless communication, it is practically impossible to eavesdrop link layer communication for a long period without errors. Although any single dynamic secret might be known to the adversary, having a complete knowledge about an entire series of dynamic secrets is extremely difficult.

This paper differs from previous research on the following aspects. Instead of working with the physical layer channel model to calculate the secrecy capacity, we shift attention to the link layer and emphasize the dynamics of secrets. This allows the implementation flexibility to integrate with existing security mechanisms. Previous research focuses on the perfection of secrets. In this paper, we propose to XOR many not-so-perfect dynamic secrets together to increase security strength. When a system secret is leaked, the ongoing communications can gradually replenish the security by generating new dynamic secrets. The conventional goal is to generate an impeccable secret and then take on the burden to carefully protect it. In our scheme, it is the adversary that need continuous efforts to track the dynamically changing system secret.

III. Secret Safety Model

In this section, we use a qualitative model to compare the secret safety over time with and without the dynamic secrets. There are two wireless devices in communication. A secret key is shared in between to keep the communication secure. This secret key is the system secret in this model. The secret safety is measured by H_k, the entropy of this key conditioned on the adversary's knowledge. H_k indicates how many bits the adversary needs to guess about the key. When $H_k = 0$, the adversary knows the key explicitly and the communication is not secure.

The key has L independent random bits. In the beginning, the key is perfectly secret, $H_k = L$. As shown in Figure 1

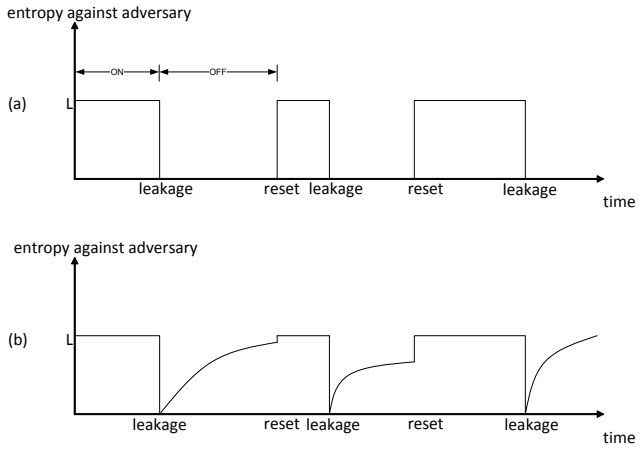

Fig. 1. time varying secret key entropy conditioned on adversary's knowledge: (a) without dynamic secrets, (b) with dynamic secrets

(a), key leakages occasionally happen. Each leakage grants the adversary complete knowledge of the secret key. There is an administrator who manually resets the secret key. Without the administrator, one leakage would compromise the security for a long period.

Even with key resets, the time interval without security can be substantial. Assuming that leakages are independent events which form a Poisson process with parameter λ_i, the administrator periodically resets the key with period T, this scenario can be modeled as an ON-OFF process (Figure 1 (a)).

The OFF period corresponds to the interval during which the security vanishes. The expected OFF time, $\overline{T_{OFF}}$ characterizes the average length of an OFF interval. By the memoryless property of the leakage process we have,

$$\overline{T_{OFF}} = \frac{T}{1 - e^{-\lambda_i T}} - \frac{1}{\lambda_i}. \tag{1}$$

Suppose that the administrator diligently resets the key once a month, and on average the secret key leakages occur once per year, e.g. $1/\lambda_i = 12T$. (1) shows that one leakage on average grants the adversary a half month for malicious activity ($\overline{T_{OFF}} \approx 0.51T$). Calculation with practical parameters demonstrates that, periodically resetting the key cannot effectively prevent damage from key leakages.

If the administrator is less regular and key resets also form a Poisson process, the expected time window for malicious activities would be even longer. Let $1/\lambda_a$ be the average time between two key reset events which corresponds to T in (1). We have

$$\overline{T_{OFF}} = \frac{1}{\lambda_a}. \tag{2}$$

With dynamic secrets, the adversary's data losses, receiving errors and other uncertainties will accumulate and increase the conditional entropy of the secret key. Therefore the vanished security can be replenished as shown in Figure 1(b). Because the adversary's information loss is hard to model and XOR operation obstructs the entropy calculation, we can only qualitatively analyze this accumulation process.

At time t during the OFF period, the secret key is $k(t)$, its entropy conditioned on the adversary's knowledge E is

$$H_k(t) = H(k(t)|E(t)). \tag{3}$$

At time $t' > t$ within the same OFF period, a dynamic secret $s(t')$ is generated. The secret key is updated by

$$k(t') = k(t) \oplus s(t') \tag{4}$$

$$H_k(t') = H(k(t) \oplus s(t')|E(t')) \geq H_k(t). \tag{5}$$

(5) shows that $H_k(t)$ is monotonically non-decreasing. If the adversary only has incomplete knowledge about the dynamic secret, $s(t')$, then it is extremely likely that $H_k(t + \delta t) = H_k(t)$.

For an active wireless data communication, such as over WiFi, it is difficult to have perfect eavesdropping without frame losses even for a short time span of several minutes. Moreover, a data frame usually contains more entropy than a secret key. One data frame loss for the adversary can fully refill the secret key entropy. Therefore the communication can be quickly re-secured when compared with the T_{OFF} calculated in (1) and (2).

As shown in Figure 1(a), communication security is either perfect or absent. When the key is not stolen by the adversary, the communication is safe. On the other hand if the adversary gains full knowledge about the key, the security is completely gone. Moreover the users are usually not aware of the fact that the key has been stolen. Our proposed scheme changes the situation to Figure 1(b). Even if the secret key is stolen, security will be regained from the communication errors in a very short period of future communications. We call this "replenishing of secrecy".

In practice, the adversary could plant a rootkit or trojan into a networked device. If the adversary can establish a link to directly retrieve unsecured data and have full control over the device, no security mechanism will work. Such a complete security breach is highly intrusive and susceptible to detection because the victim device's behavior is manipulated. In wireless communications, it is more often that the adversary steals the system secret then uses the secret to decrypt the eavesdropped data or inject malicious communications. In such circumstances, using dynamic secrets significantly restricts the adversary.

If an eavesdropping adversary only obtains the system secret by chance, the secret updates cannot be tracked for long due to the inevitable errors in wireless eavesdropping. If the adversary finds a repetitively exploitable vulnerability, dynamic secrets will force the adversary to frequently use it because eavesdropping cannot follow the secret updates. Therefore the adversary's risk is substantially increased. Moreover, any malicious injection will be detected because the malicious communication changes the system secret in one device but not the other.

Another difficult situation occurs when many eavesdroppers collude to reduce the probability of receiving error to an extremely low value. In this case, the key regains entropy slowly and dynamic secrets offer little entropy benefit (nor any harm). On the other hand, dynamic secrets will hinder the collusion attack. Adversaries constantly suffer from a single point of failure problem. Besides receiving errors, many unfavorable events can destroy the colluding scheme such as the movement of target wireless devices. There is no guarantee that colluding adversaries can maintain perfect eavesdropping all along till the sensitive data of their interests is transmitted. Before an attack, the adversary must evaluate the incentive against the soaring risk and cost to maintain multiple eavesdropping devices all the time.

IV. Extracting Dynamic Secrets

In this section, we introduce algorithms for dynamic secret extraction as well as the mathematical rationales behind them. These algorithms monitor link layer communication, especially error retransmissions, to synchronously select a group of frames for both sender and receivers. These frames are then hashed into dynamic secrets.

A. Automatic Error Tracing

Automatic Error Tracing (AET) algorithms monitor the link layer error retransmission process at both the sender and receiver. Without additional communication, the sender and receiver can select a set of frames known as one time frames (OTFs). The term "one time frame" refers to a frame that is only aired once and correctly received. OTFs are most likely to be lost or erroneously received by the adversary. We use the widely implemented Stop-and-Wait retransmission protocol (SW) as an example to demonstrate how AET works. A well-known application of SW is WLAN link layer retransmission [10].

In Figure 2, as regulated by SW, the sender transmits a frame and waits for the corresponding acknowledgement before transmitting any new frame. If the acknowledgement does not arrive before a timeout, the retransmission occurs. Figure 2 also shows how Algorithms 1 and 2 identify the one time frames.

AET algorithms are defined in Algorithm 1 and 2. The frame format contains two important fields: a retransmission flag and a sequence number. We use the postfix $.retran$ and $.serial$ to note them. Ψ_s and Ψ_r are the sender and receiver set of OTFs respectively and $\Psi_s = \Psi_r = \phi$ before communication begins.

On the sender side, if a frame is only transmitted once and its acknowledgement frame is received in time, this frame is added to the one time frame set Ψ_s. As shown in Figure 2, frames 1, 4 and 5 are added to Ψ_s because their acknowledgements arrive before a timeout. Frames 2 and 3 on the other hand experience retransmissions and do not belong to Ψ_s.

The receiver identifies an one time frame by its immediate successor frame. If the current frame is not retransmitted, and the next frame received is a new frame with different sequence number, then the current frame is identified as an OTF and is added to Ψ_r. Frame 1 is identified as OTF when the re-transmitted frame 2 arrives. Frame 4 is identified when frame 5 arrives. This process continues. Frames 2 and 3 in Figure 2 are recognized as non-OTF because their re-transmission flags are set.

Algorithm 1: AET sender

```
foreach frame m_i do
    m_i.retran = 0;
    send m_i;
    while true do
        wait on ACK or time out;
        if ACK received then
            Jump out the loop;
        m_i.retran = 1;
        send m_i;
    if m_i.retran = 0 then
        Add m_i to Ψ_s;
```

Algorithm 2: AET receiver

```
foreach received frame m_i do
    if m_i integrity check pass then
        send ACK;
        if m_i.serial ≠ m_{i-1}.serial, m_{i-1}.retran = 0
        then
            Add m_{i-1} to Ψ_r;
```

Ψ_s and Ψ_r are automatically synchronized by the sender's and receiver's local information. No additional communication is needed.

The sender explicitly knows which frames are OTFs. The receiver could identify whether a frame is OTF or not as soon as a new frame is received. Ψ_r differs from Ψ_s by at most one frame. After Ψ_s reaches a certain size threshold n_{ts} by adding a frame m_i, the sender sends frame m_{i+1} to the receiver. The acknowledgement of m_{i+1} confirms that the receiver had added m_i to Ψ_r and $\Psi_r = \Psi_s$.

Algorithms in the following sections will only operate on the synchronized Ψ_s and Ψ_r. Therefore, we unify the notion as

$$\Psi = \Psi_s = \Psi_r. \tag{6}$$

It is very difficult for the adversary to reproduce Ψ. Not only must she eavesdrop every data frame, but also all of the acknowledgements and retransmissions. Otherwise even if she can receive a frame correctly, she cannot identify whether this frame is an OTF or not. Whenever the adversary is uncertain about Ψ, the uncertainty is reflected in the dynamic secrets.

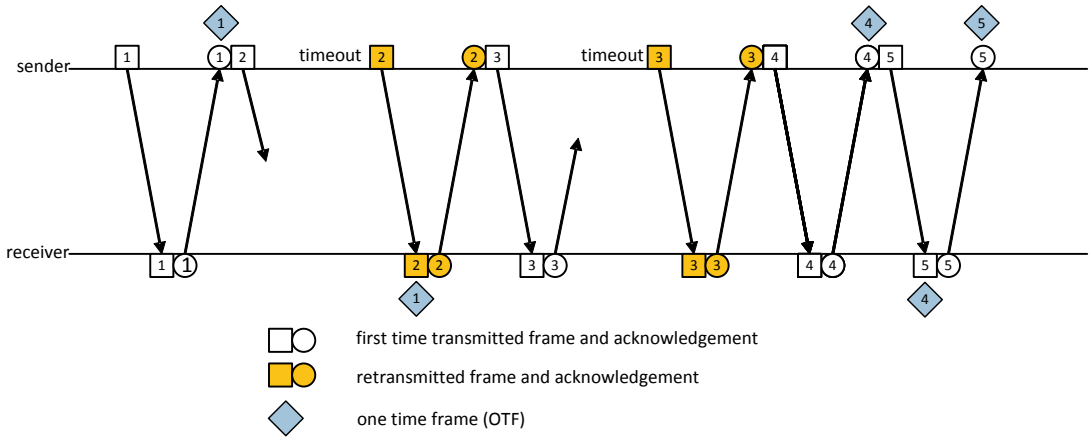

Fig. 2. AET working with SW protocol

B. Extracting Information Privilege

After automatically selecting the one time frames for both sender and receiver, the immediate task is to generate a dynamic secret from Ψ while retaining as much of the adversary's information loss as possible. The problem is, we do not know what is actually lost by the adversary.

The theory of random hashing provides a solution. Regardless of the actual input distribution, by uniform-randomly choosing a function from a universal-2 hashing class, the expected hash output distribution will be close to the uniform distribution when the output is sufficiently short [7]. This result can be applied to the OTF set Ψ. When Ψ is hashed to a short hash value S, the conditional distribution of S given the adversary's knowledge can be close to the uniform distribution. Because a nearly uniform distribution accounts for nearly maximum entropy, the adversary knows almost nothing about S.

Formally, let $E = e$ represent the adversary's knowledge about Ψ. Regardless of the actual conditional distribution $p(\Psi|E = e)$, by uniformly randomly choosing from a class of universal-2 hash functions, the conditional distribution of hash output S is close to uniform when S is sufficiently short. The choice of hash function can be known by the adversary. When averaged over all random choices of hash functions, we have

$$p(S = s|E = e) \approx \frac{1}{|\{S\}|} \quad \forall\, s \in \{S\} \tag{7}$$

$$H(S|E = e) \approx \log_2 |\{S\}|. \tag{8}$$

Most of the time, we cannot predict how short S needs to be. Instead, we rely on the more generalized result from [4]. This result shows the analytical relationship between the length of generated secret, the bound on adversary's information loss, and the conditional entropy.

Let a binary string X represent the common information Ψ. E represents adversary's knowledge about X. E only contains incomplete information about X. The information loss is bounded by $H_2(X|E = e) \geq \epsilon$, where $H_2(\cdot)$ is the Rényi entropy of order 2 [20].

Privacy Amplification: Let $\mathcal{F} : GF(2^{l_x}) \rightarrow GF(2^{l_s})$ be a universal-2 hashing family defined in [7] where l_x is the length of X and l_s is the length of S. X and E are related by $H_2(X|E = e) \geq \epsilon$ where ϵ is a positive value. F is the random variable corresponding to a uniformly random choice from $\mathcal{F}$. Let $S = f_{PA}(X)$, where $f_{PA}(\cdot)$ is the uniformly random choice from $\mathcal{F}$, then

$$\begin{aligned} H(S|F, E = e) &\geq H_2(S|F, E = e) \\ &\geq l_s - \log_2 (1 + 2^{l_s - \epsilon}) \\ &\geq l_s - \tfrac{2^{l_s-\epsilon}}{\ln 2} \end{aligned} . \tag{9}$$

When the probability is at least $1 - \delta$ that E takes value e such that $H_2(X|E = e) \geq \epsilon$, (9) can be generalized as

$$H(S|F, E) \geq (1 - \delta)(l_s - \log_2 (1 + 2^{l_s - \epsilon})). \tag{10}$$

When $l_s \leq \epsilon$, S is almost perfectly secret because averagely Eve would have less than one bit information about S. The second inequality in (9) shows that even if $l_s > \epsilon$, the adversary's expected uncertainty toward S is bounded from below by $\epsilon - 1$.

$$H(S|F, E = e) \geq l_s - \log_2 (1 + 2^{l_s - \epsilon}) \geq l_s - (1 + l_s - \epsilon) = \epsilon - 1 \tag{11}$$

(11) shows that even S have more bits than the adversary's information loss, the information loss is almost fully retained in S.

It's noteworthy that (9) and (10) are averaged over all uniformly random choices of hash functions. There is a non-zero probability that when $H_2(S|E = e) \geq \epsilon$ and $l_s \leq \epsilon$, for some specific values of F, $H(S|F, E = e)$ is not negligible. However, these combinations of S, F and e only appear with negligible probability [4].

The above discussion shows that, by utilizing universal-2 hashing, the adversary's information loss will be almost fully retained in the hash value regardless of its length. Algorithm 3 compresses Ψ into a dynamic secret by universal-2 hashing. Once the number of OTFs in Ψ reaches a threshold n_{ts}, Sender and receiver agree on a uniformly random choice of universal-2 hash functions, $f_{PA}(\cdot)$, to compress Ψ into the

Algorithm 3: dynamic secret extraction

```
if |Ψ| ≥ n_ts then
    agree on a randomly chosen function f_PA from a
    universal-2 hash function class;
    s(t) = f_PA(Ψ);
    Ψ = φ;
```

dynamic secret $s(t)$. The frames in Ψ are concatenated into a long binary string to be hashed. After hashing, Ψ is reset to the empty set. As communication goes on, algorithms 1, 2 and 3 repeatedly create a series of dynamic secrets.

Algorithm 3 does not require any prior secret shared between legitimate users. Nothing is lost by allowing the adversary to know the selection of hash functions. It is possible that few poor choices of hash functions for some specific Ψ would allow the adversary to know a non-negligible portion of $s(t)$ while his information loss to Ψ is large. However, as indicated in [4], the bad combination only occurs with negligible probability.

The bits in Ψ need not to be independent and or ideally random because (9) and (10) hold for arbitrary distributions.

C. Generalization

Other reliable communication mechanisms can also apply the same framework to generate dynamic secrets. The above algorithms need to be modified because many retransmission protocols cannot automatically trace errors and select OTFs without additional communication, such as Go-Back-N ARQ and Selective Repeat ARQ. For these protocols, it is necessary for the receiver to maintain a buffer, namely Ψ'_r. Ψ'_r contains frames that have only been received once. The sender explicitly knows the sending sequence and therefore has no difficulty maintaining the OTF set Ψ_s. The following relationship is always true.

$$\Psi_s \subseteq \Psi'_r \tag{12}$$

When $|\Psi_s| = n_{ts}$, the sender notifies the receiver of Ψ_s's index set. The receiver then selects the corresponding OTFs from Ψ'_r and forms $\Psi_r = \Psi_s = \Psi$. A dynamic secret is then generated using Algorithm 3.

There are fast and efficient implementations of universal-2 hash functions [11]. If the complexity is still a burden for some wireless devices, $f_{PA}(\cdot)$ can be replaced with a strong, deterministic hash function $f_H(\cdot)$ such as SHA-256 [19]. A hash function spreads one input bit's uncertainty to many bits in the output. It's hard to obtain a close form bound on adversary's information loss regard to $s(t)$ due to the complex structure of these hash functions. On the other hand, the collision resistance property of hash functions also prohibit the adversary from exploiting information advantage.

Assume the adversary has partial information about Ψ. Given this information, Ψ can only take values from a set $\mathcal{X}$. $\mathcal{X}$ is a subset of all feasible values of Ψ. The hash values forms another set $\mathcal{S}$,

$$\mathcal{S} = \{s | s = f_H(x), \forall\ x \in \mathcal{X}\}. \tag{13}$$

Because $f_H(\cdot)$ is a many-to-one mapping, it is possible that $|\mathcal{S}| < |\mathcal{X}|$ and $H(\mathcal{S}) < H(\mathcal{X})$. In such cases, the adversary has less uncertainty about the dynamic secret than about Ψ. However, the hash function is designed to be collision resistant. It is hard to find $x_1, x_2 \in \mathcal{X}$, $x_1 \neq x_2$ and $f_H(x_1) = f_H(x_2)$. Therefore, to obtain $\mathcal{S}$, the adversary must calculate $f_H(x)$ for every $x \in \mathcal{X}$ as if $H(\mathcal{S}) = H(\mathcal{X})$ and the information loss about Ψ is fully retained in the dynamic secret.

V. Protecting System Secrets

It is possible that the adversary perfectly receives all the frames and acknowledgements in a period when Ψ grows from ϕ to size n_{ts}. In this case, the adversary can determine the corresponding dynamic secret $s(t)$. A single dynamic secret may not be secure enough. However, when a series of dynamic secrets are generated and XORed, very likely the adversary will lose information about some of these dynamic secrets, and **the information loss will accumulate**. In this section, we discuss how to apply a series of dynamic secrets $\{s(t)|t = t_0, t_1, \cdots\}$ to protect the system secret in wireless communications.

Let T be the set of times when dynamic secrets are created.

$$T = \{\tau | \textit{a dynamic secret } s(\tau) \textit{ is created}\} \tag{14}$$

For $\tau_1, \tau_2 \in T$ and $\tau_1 \neq \tau_2$, $s(\tau_1)$ and $s(\tau_2)$ are independent. This is because the choices of $f_{PA}(\cdot)$ at τ_1 and τ_2 are independent. For arbitrary inputs, the uniformly random choices of hash function evenly distribute the hash output over all possible values. Even two OTF sets, $\Psi(\tau_1)$ and $\Psi(\tau_2)$, are known to be identical, the corresponding hash output $s(\tau_1)$ and $s(\tau_2)$ are still independent because the two random choices of $f_{PA}(\cdot)$ are independent to each other and values of Ψ.

The dynamic secret is divided into two sequences of bits $u(t)$ and $v(t)$, $s(t) = u(t)||v(t)$, to protect the private-public key pair and secret symmetric key respectively. The length of $u(t)$ can be customized according to a specific security requirement. $v(t)$ is the same length as the symmetric encryption key. Let $U(t)$ and $V(t)$ be

$$U(t) = \bigoplus_{\tau \in T,\ 0 \leq \tau \leq t} u(\tau) \tag{15}$$

$$V(t) = \bigoplus_{\tau \in T,\ 0 \leq \tau \leq t} v(\tau) \tag{16}$$

As time t elapses, $U(t)$ and $V(t)$'s entropy and conditional entropy given the adversary's knowledge E will be maximized.

$$H(U(t)|E) \quad \rightarrow H(U(t)) \quad \rightarrow |U(t)| \tag{17}$$

$$H(V(t)|E) \quad \rightarrow H(V(t)) \quad \rightarrow |V(t)| \tag{18}$$

Let $k(t)$ represent the secret symmetric key. We propose the following algorithm to dynamically update $k(t)$ using $v(t)$.

Algorithm 4: symmetric key protection

```
if a new s(t) is generated then
    k_tmp = k ⊕ v(t)
    if k_tmp is not a weak key then
        k(t) = k_tmp
```

Whenever a new dynamic secret $s(t)$ is generated, Algorithm 4 attempts to update the encryption key $k(t)$ by XORing it with $v(t)$. It is necessary to check whether this operation will produce a weak key for the symmetric cipher. For example, when $v(t) = k(t)$, the new encryption key is a string of all 0s. The all 0 key in many symmetric ciphers can cause the ciphertext and the plaintext to be equivalent. When a weak key is detected, the algorithm skips the update. The decisions on whether or not to update the encryption key are inherently synchronized by the AET and SW error retransmission protocol. No additional communication is needed.

Each execution of Algorithm 4 will enhance the protection. Because $k(t)$ and $v(t)$ are independent, the entropy is non-decreasing,

$$H(k(t) \oplus v(t)) \geq \max\{H(k(t)), H(v(t))\} \tag{19}$$

The entropy increment in most cases is strictly positive.

Recall the secret safety model in Section III, (19) illustrates how the security gap is replenished by subsequent communications. Suppose at time t_0, the encryption key is leaked to the adversary, and wireless communication continues,

$$H(k(t_0)|E) = 0 \tag{20}$$

At time $t_1 > t_0$, the encryption key $k(t_1)$ is

$$k(t_1) = k(t_0) \oplus \bigoplus_{\tau \in T, t_0 < \tau \leq t_1} v(\tau). \tag{21}$$

The adversary's uncertainty about $k(t_1)$ is lower bounded as

$$\begin{aligned} H(k(t_1)|E) &\geq \max\{H(k(t_0)|E), H(\bigoplus_{\tau \in T, t_0 < \tau \leq t_1} v(\tau)|E)\} \\ &= \max\{0, H(\bigoplus_{\tau \in T, t_0 < \tau \leq t_1} v(\tau)|E)\} \\ &= H(\bigoplus_{\tau \in T, t_0 < \tau \leq t_1} v(\tau)|E) \end{aligned} \tag{22}$$

When there are many updates between t_0 and t_1, the adversary will gradually lose information about the key. In typical data wireless communications, a data frame contains much more randomness than a encryption key. If the adversary lost a single OTF m_i between t_0 and t_1, $m_i \in \Psi(\tau_i)$ and $s(\tau_i)$ is generated from $\Psi(\tau_i)$, then $v(\tau_i) \in s(\tau_i)$ will be almost completely random to the adversary. In such cases, we have

$$H(k(t_1)|E) \geq H(\bigoplus_{\tau \in T, t_0 < \tau \leq t_1} v(\tau)|E) \approx |k(t_1)|. \tag{23}$$

The encryption key $k(t)$ can be re-secured with a single frame loss by the adversary.

As noted in Algorithm 4, possible conflicts between dynamic secrets and encryption algorithms should be analyzed. The protection provided by the dynamic secrets can be made vulnerable if the encryption function is poorly designed. For example, the symmetric key encryption function $e_k(\cdot)$ is linear respect to the exclusive OR operation in the key space.

$$e_{k_1 \oplus k_2}(x) = e_{k_1}(x) \oplus e_{k_2}(x) \tag{24}$$

Let k_1 and $k_2 = k_1 \oplus v(t)$ be two consequent session keys. $v(t)$ is known by the adversary because of the perfect eavesdropping of frames which are used to create $s(t)$. When the same plaintext x is encrypted by both k_1 and k_2, the corresponding ciphertexts c_1 and c_2 are

$$\begin{aligned} c_1 &= e_{k_1}(x) \\ c_2 &= e_{k_2}(x) = e_{k_1 \oplus v(t)}(x) \end{aligned} \tag{25}$$

Then the adversary can decrypt x by

$$x = d_{v(t)}(c_1 \oplus c_2). \tag{26}$$

Such a trivial vulnerability is unlikely under modern cryptographic standards. Non-linear modules are key components and difference makers in symmetric cipher design [21]. To further reduce the concern that the XOR operation might interfere with the encryption algorithm, we can replace XOR in Algorithm 4 with decryption function $d_k(\cdot)$ in the symmetric cipher. Then,

$$k_{tmp} = d_k(v(t)). \tag{27}$$

There is no close form bound to guarantee the entropy increment between successive session keys due to the complexity of $d_k(\cdot)$. However, the key at least continues to change and can defeat attacks that require a large quantity of data encrypted by the same key.

When $v(t)$ is known by the adversary, deriving the new encryption key value k_{tmp} is equivalent to recovering the plaintext by ciphertext only, e.g. breaking the symmetric cipher. When the adversary does not have complete information about $v(t)$, deriving k_{tmp} will be more difficult. The session key $d_k(v(t))$ is at least as secure as the cipher itself. It is more secure when the adversary does not have perfect eavesdropping records.

To protect the private key in a public key infrastructure, we need an additional symmetric encryption layer instead of directly operating on the private key, as shown in Figure 3. Once the adversary loses information about $U(t)$, even if the private key is stolen, the data is still secure.

VI. Bootstrapping Wireless Security

As analyzed in the previous sections, dynamic secrets can accumulate information loss at the adversary. In some scenarios, it is even possible to bootstrap wireless security with dynamic secrets.

The sender and receiver publicly agree on a specific starting value, such as all 0s, as the initial system secret. Then the sender keeps transmitting random data and waits for the security strength to build up. The sender then sends sensitive

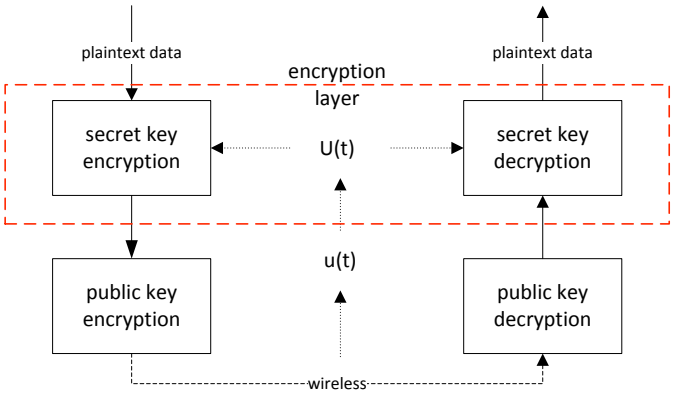

Fig. 3. protect public key infrastructure

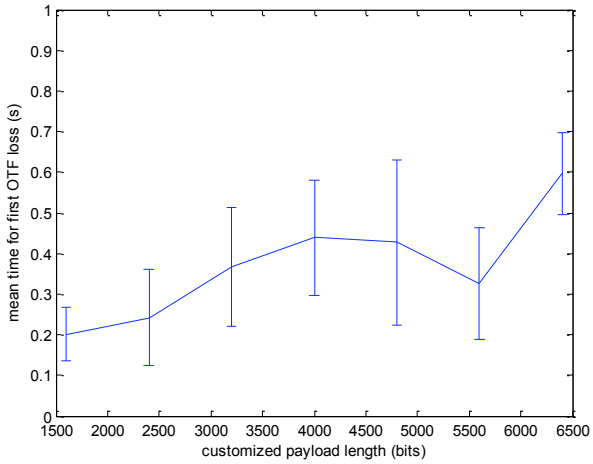

Fig. 4. office environment bootstrapping measurement

data only after it believes the system is sufficiently secure. From another perspective, the bootstrapping process never stops until sensitive data is transmitted.

If the adversary does not eavesdrop from the beginning, the information loss is not recoverable. Many unpredictable events such as transmission errors or user mobility cause information loss as well. Our scheme could be an alternative for devices that communicate with each other for a long time and are unwilling to invest in complex public key infrastructure.

For example, in enterprise wireless LAN, wireless adaptors and the access point can bootstrap secure communication with dynamic secrets. A wireless LAN provides sufficient bandwidth to quickly build up the security. Users can intentionally generate non-sensitive wireless traffic such as through web browsing or sharing some random files before transmitting security sensitive information. Moreover, when a user comes to the office and reconnects to the enterprise wireless LAN, all his previous communications had contributions in the entropy of current system secret.

We prototyped an enterprise wireless LAN bootstrapping scenario in our lab. Two wireless nodes attempt to bootstrap a 256 bit symmetric key against a "curious" adversary who uses a laptop and commercial-available wireless adaptor to eavesdrop. Because a wireless LAN frame is much longer than the key, the sender believes the loss of one OTF is sufficient to defeat the adversary.

In our experimental setting, we first fix the location of two wireless users, then test the frame loss rate on several different locations. We deploy the adversary in the location with the lowest frame loss rate.

Figure 4 shows that the adversary loses the first one time frame within a second. A conservative user can transmit 10 seconds of random data to the receiver before normal communication. This would defeat the adversary with overwhelming probability.

VII. Prototype Implementation

We build a prototype over a wireless LAN. The hardware of the wireless nodes are computers with USB 802.11g adaptors. We program in user space with Linux using a default kernel driver. There are several reasons for this architecture. Wireless LAN is one of most widely deployed wireless data communication frameworks. It uses SW for link layer retransmission. We implement algorithms in user space. Our experience suggests that it it straightforward for application developer to include dynamic secrets without kernel modification.

SW is re-implemented in our prototype using broadcast frames. Broadcasting frames will not trigger hardware controlled retransmissions. The payload size of each frame is limited to avoid link layer fragmentation. We use wireless sniffing to verify that there is no link layer retransmission occurs.

Figure 5 illustrates the work flows for the prototype. When the sender's OTF set Ψ_s reaches the threshold n_{ts}, a frame that contains random choice of hash function is sent to the receiver. This particular frame also fills up the receiver's OTF set Ψ_r by pushing the prior arrived frame into it.

This particular hash frame is the frame that ensures $\Psi_s = \Psi_r$ in Section IV. It also tells the receiver about the sender's choice of universal-hash function. The sender must wait for the acknowledgement of the hash frame before it can extract the dynamic secret using Algorithm 3. The receiver can extract the secret once it receives the hash frame and the hash function choice is known. As introduced in Section IV-C, when a deterministic hash function is used instead of universal hash functions, the work flows in Figure 5 remains unchanged. The only difference is that the hash frame is replaced by normal data frame.

VIII. Summary and Conclusion

We propose a scheme to secure wireless communications when the underlying system secret can be stolen. Our solution is to generate a series of hash values from the communication process, namely *dynamic secrets*, and apply these hash values to constantly update the system secret.

Associating system secrets with the communication process imposes a single point of failure problem to the adversary. In

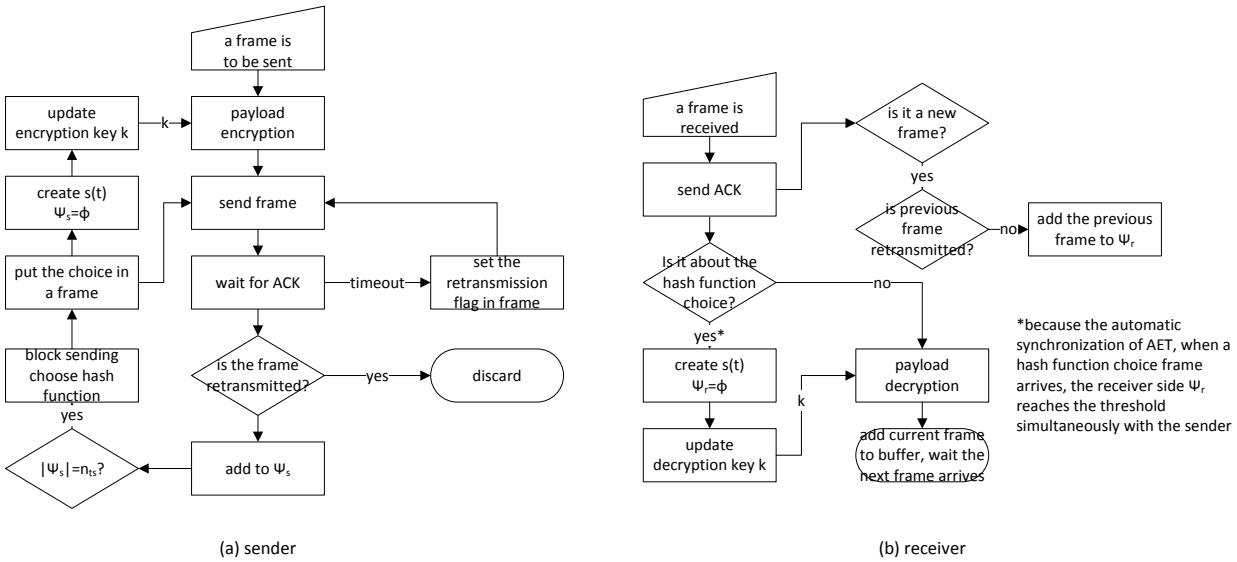

Fig. 5. prototype work flows: (a) sender, (b) receiver

the error-prone wireless communications, the adversary must maintain eavesdropping without information loss to trace the system secret updates. Moreover, the adversary's information loss can accumulate and the communication security is improved over time. As communication goes on, communication security keep replenishing itself as if the wireless communication system has elasticity to resist attacks.

ACKNOWLEDGMENT

This work is supported in part by National Science Foundation under grants CNS-0519922, CNS-0524323, CNS-0721790, EFRI-0735974, and DARPA under Contract W911NF-08-1-0233.

REFERENCES

[1] T. Aono, K. Higuchi, T. Ohira, B. Komiyama, and H. Sasaoka. Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. *Antennas and Propagation, IEEE Transactions on*, 53(11):3776–3784, Nov. 2005.

[2] Babak Azimi-Sadjadi, Aggelos Kiayias, Alejandra Mercado, and Bulent Yener. Robust key generation from signal envelopes in wireless networks. In *CCS '07*, pages 401–410, 2007.

[3] Joao Barros and Miguel R. D. Rodrigues. Secrecy capacity of wireless channels. *Information Theory, 2006 IEEE International Symposium on*, pages 356–360, July 2006.

[4] Charles H. Bennett, Gilles Brassard, Claude Crkpeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transaction on information theory*, 41:1915–1923, 1995.

[5] M. Bloch, J. Barros, M. R. D. Rodrigues, and S. W. McLaughlin. Wireless information-theoretic security - part i: Theoretical aspects. *IEEE Trans. on Information Theory*, 2006.

[6] M. Bloch, J. Barros, M. R. D. Rodrigues, and S. W. McLaughlin. Wireless information-theoretic security - part ii: Practical implementation. *IEEE Trans. on Information Theory*, 2006.

[7] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:396–407, 1979.

[8] Olivier Chevassut, Pierre alain Fouque, Pierrick Gaudry, and David Pointcheval. Key derivation and randomness extraction. Technical report, In Crypto 05, 2005.

[9] Amer A. Hassan, Wayne E. Stark, and John E. Hershey. Cryptographic key agreement for mobile radio. *DSP*, 6:207–212, 1996.

[10] IEEE-SA. Ansi/ieee standard 802.11, part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Technical report, IEEE-SA Standards Board, 2003.

[11] Hugo Krawczyk. Lfsr-based hashing and authentication. In *CRYPTO '94*, pages 129–139, London, UK, 1994. Springer-Verlag.

[12] Ruoheng Liu, Tie Liu, H. Vincent Poor, and Shlomo Shamai. Multiple-input multiple-output gaussian broadcast channels with confidential messages. *CoRR*, abs/0903.3786, 2009.

[13] Ruoheng Liu and H.Vincent. Poor. Multi-antenna gaussian broadcast channels with confidential messages. In *ISIT 2008. IEEE International Symposium on Information Theory, 2008.*, 2008.

[14] Suhas Mathur, Wade Trappe, Narayan Mandayam, Chunxuan Ye, and Alex Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *MobiCom '08*, pages 128–139, 2008.

[15] U. M. Maurer. Secret key agreement by public discussion from common information. *IEEE Trans. on Information Theory*, 39:733–742, 1993.

[16] U. M. Maurer and S. Wolf. Secret key agreement over a non-authenticated channel - part i: Definitions and bounds. *IEEE Transactions on Information Theory*, 49:822–831, 2003.

[17] U. M. Maurer and S. Wolf. Secret key agreement over a non-authenticated channel - part ii: The simulatability condition. *IEEE Transactions on Information Theory*, 49:832–838, 2003.

[18] U. M. Maurer and S. Wolf. Secret key agreement over a non-authenticated channel - part iii: Privacy amplification. *IEEE Transactions on Information Theory*, 49:839–851, 2003.

[19] NIST. Secure hash signature standard (shs) (fips pub 180-2). Technical report, NIST, 2001.

[20] Alfred Renyi. On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, 1960.

[21] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[22] Xiaojun Tang, Ruoheng Liu, Predrag Spasojevic, and H. Vincent Poor. On the throughput of secure hybrid-arq protocols for gaussian block-fading channels. *CoRR*, abs/0712.4135, 2007.

[23] Luca Trevisan. Extractors and pseudorandom generators.

[24] R. Wilson, D. Tse, and R.A. Scholtz. Channel identification: Secret sharing using reciprocity in ultrawideband channels. In *ICUWB 2007*, pages 270–275, Sept. 2007.

[25] Chan Wong Wong, J.M. Shea, and T.F. Wong. Secret sharing in fast fading channels based on reliability-based hybrid arq. In *MILCOM 2008. IEEE*, pages 1–7, Nov. 2008.

[26] Sheng Xiao, Hossein Pishro-Nik, and Weibo Gong. Dense parity check based secrecy sharing in wireless communications. In *Proceedings of Globecom'07*, 2007.