A Design of Secure and Reliable Wireless Keyboards and Mice Against Man-In-The-Middle Attacks\textsuperscript{1}

Lake Bu, \textit{Student Member, IEEE}, Mark Karpovsky, \textit{Life Fellow, IEEE}

Reliable Computing Laboratory, Electrical and Computer Engineering
Boston University, Boston, USA

\textbf{Abstract}—It has been publicly known that keyboards and mice, either wired or wireless, can be potentially attacked by eavesdropping or hijacking. Although there are many secure keyboards and some secure mice in the market, there are still vulnerabilities under different types of Man-In-The-Middle (MITM) attacks such as hijack or replay. Usually these attacks are low cost and easy to implement, but so far there have not been any effective solutions to them. This paper proposes a protection scheme against the existing and potential MITM attacks to wireless keyboards and mice while keeping a low overhead in hardware and energy consumption, which is critical to battery powered devices. This lightweight scheme can also be applied to other wireless devices sensitive to battery life such as implantable medical devices.

\textbf{Index terms} — keyboard, mouse, security, eavesdropping, hijack, tampering, replay, man-in-the-middle, encryption, authentication, AES, AMD, error control codes.

\section{I. INTRODUCTION}

Wireless keyboard and mouse security has always been a repeatedly reported issue in the recent years. Since the keyboard and mouse transmission protocol, scancodes and packet formats are accessible to everyone, it is not too hard to find its vulnerabilities and apply corresponding attacks.

Eavesdropping (or keylogging) is one of the most commonly seen attacks to wireless keyboards. There are software and hardware designed to listen and record the scancodes, and then translate them to readable key strokes for attackers. In the past few years, a couple of criminal cases have been reported that attackers used hardware key loggers to steal other people’s important information such as account names and passwords. In 2013, several U.S college students were caught installing hardware key loggers and eavesdropping their professors’ keyboard activities. By the acquired account information they either stole the exam papers or even directly changed their grades from F’s to A’s [1]. Hardware key loggers are usually installed in the transmission channel between the keyboard and the computer. For wired keyboards they are plugged in the USB slots, and for wireless keyboards on walls’ electricity outlets. They are favored by many hackers because they are hard to be noticed and not expensive at all. Software key loggers have also been utilized to compromise computers. In March 2016 it was reported that attackers used a key logger program named “Olympic Vision” to steal business email accounts of companies of 18 countries.

Besides passive eavesdropping, attackers can also intercept the transmitted keystrokes and mouse movements in order to spoof or modify them to malicious messages. These active attacks also belong to “Man-In-The-Middle (MITM)” attacks. Starting from 2016, mouse hijack has been made known in the security community for its easy implementation and high success rate. By a less than $15 wireless dongle attackers can spoof the victim’s mouse click and then gain access to the victim’s computer. Billions of devices are exposed to this attack [2]. There is also a risk that an attacker may be able to tamper a keyboard’s scancodes to other wanted values, say changing “You owe me $1000” to “You owe me 9000” by altering only one byte of data. For even worse cases, even if the devices are encrypted, attackers can still store the legal ciphers and replay them again to spoof the victim’s operation.

So far many keyboard and mouse manufacturers have designed their secure human interface devices (HID) with AES encryption to protect users from passive eavesdropping [3]. However, although AES alone makes the transmission unreadable to attackers, it does not resist any active MITM attacks. Because of the vulnerability in AES’ encryption procedure, MITM attackers can still hijack or tamper the transmissions.

Therefore in this paper we propose a design of secure wireless keyboards and mice using authenticated encryption against both passive and active MITM attacks, which to our knowledge no other secure HID manufacturers in the market have ever provided. The major contributions of this design are:

- It uses encryption against eavesdropping. It also randomizes the plaintext to avoid one-on-one mapping between HID operations and their ciphers;
- It checks the authenticity of each HID transmission to verify if it has been tampered. In the experiments it has missed not a single attack in all the transmissions during the lifespan of a device;
- All encrypted legal HID messages have their timestamps, so that a prior legal message cannot be stored and replayed ever again during the lifespan of a device;
- It provides more reliability than current HIDs by enabling double random error correction;
- Its two modes’ energy consumption is only 4\% and 13\% respectively of the conventional method’s.

The rest of the paper is organized as the following. Section II briefly explains the HID transmission protocols. Section III illustrates the existing and potential attack models against current secure HIDs. Section IV explains the criteria of the protection against such attacks. Section V introduces the proposed protection scheme and its work flow, as well as the theoretical estimation of its security level. Section VI evaluates the proposed design by experiments and overhead comparison with other possible schemes.

\textsuperscript{1}This work was sponsored by the NSF grant CNS 1012910.
II. WIRELESS KEYBOARD AND MOUSE PROTOCOLS

Today most wireless keyboards and mice use 2.4GHZ RF transceivers in the communication with computers. Their transmission packet format might be slightly different among manufacturers. However the information portion of the packets are always similar.

For wireless keyboards there are 2 bytes \((2 \times 8 = 16\) bits) carrying the keystroke information as shown in the following figure. The first byte indicates whether this is a pressing or releasing of a key. The second byte represents the scancode of the corresponding key. Lookup tables of the keys, scan codes, and ASCII codes can be found online or from each keyboard manufacturers’ manuals.

For wireless mice there are more information bits transmitted every time. There are 4 bytes \((4 \times 8 = 32\) bits) to represent mouse movements of \(x\), \(y\), and \(z\)-displacements (some mice do not have the \(z\)-displacement).

III. EXISTING AND POTENTIAL ATTACKS

For wireless keyboards and mice, eavesdropping and hijack are two most frequently reported attacks. As mentioned in the introduction section, secure HID manufacturers such as Microsoft and Logitech proclaim that they have equipped their wireless keyboards (a few of them mice) with 128-bit AES encryption. Although this prevents the attackers from eavesdropping and understanding the keystrokes, the devices can still be vulnerable in certain ways under active MITM attacks.

According to the existing keylogging and mouse hijack attacks, they usually have the following preconditions:
1) The attacker is able to eavesdrop the victim’s wireless HID transmission;
2) The attacker is physically close enough to the victim’s computer screen to peep at the HID inputs, at least some of them;
3) The attacker is able to use his/her own dongle to send forged mouse clicks or keystrokes to the victim’s computer.

Given the assumptions above, the following sub-sections will describe the existing and potential attacks to wireless HIDs, and even secure HIDs.

A. Eavesdropping (keylogging)

Eavesdropping is a type of passive attacks that the attacker listens to the unencrypted wireless keyboard or mouse transmission channel to acquire the keystrokes. The attacker does not necessarily apply any malicious modifications to the channel. Usually the goal of eavesdropping is to acquire the victim’s important information such as account numbers, usernames and passwords.

The Advanced Encryption Standard (AES) is a well-known solution that prevents the attackers from logging and understanding the message even if they intercept it. At present most secure HID manufactures such as Microsoft and Logitech have equipped their wireless keyboards with 128-bit AES [3] and a few also have extended this protection to their wireless mice.

B. Mouse Hijack

Mice are more vulnerable since few of them are encrypted, not to say against any active attacks. Mouse hijack has become a well known issue starting from 2016 [2], that attackers can hijack and forge mouse clicks and then spoof keyboard inputs in order to gain access to computers. Reports have said that billions of wireless devices are under this threat.

Most wireless mice are not encrypted. Even if they are, they are not properly authenticated [4]. This gives attackers opportunities to forge legal mouse clicks to spoof the legal device. As an attacker mimics mouse click packets from his/her own dongle and bypass the pairing mode of the victim’s computer, the attacker’s dongle can force-pair with the victim’s computer without the user’s permission. Then if the attacker is close enough to the victim’s computer screen, he/she will be able to send keystroke packets in order to gain full access to the victim’s computer. The attack is simple and cheap to implement by a $15 dongle and a few lines of code [5].

The affected HID manufacturers include but not limited to: Logitech, Dell, HP, Lenovo, Microsoft, Gigabyte and AmazonBasics [6].

Also since most keyboard and mouse manufacturers use one-time programmed chips for their HIDs, it is impossible to apply any firmware updates. Thus their HIDs are still under the threat of mouse hijack attacks.

C. Tampering

Although all secure keyboards have encrypted their keystroke transmission by AES, if the attacker tampers the encrypted cipher text, after decryption AES alone will not be able to verify its authentication. One of the attacks taking advantage of this vulnerability is “bit(byte)-flipping” attack. By maliciously modifying a cipher, the decrypted plain text is altered to some other content. It is especially difficult to spot when this happens to a string of numbers such as account numbers or amounts of money. Although this has not been reported, it is a potential threat to all keyboard users.

Bit(byte)-flipping attacks are commonly seen in the AES-CBC mode which most secure keyboard has been using. Even if an attacker does not have the knowledge of the secret key, as long as he/she can listen to the channel and has a proper guess of the incoming keystrokes, the attack is more likely to succeed.

According to the AES-CBC decryption procedure, a bit-flipping attack can be applied as the following [7].

Moreover, according to the preconditions 2) and 3), if an attacker is able to have the record of a few previously typed keys, it is possible to predict the incoming ones. By
the monogram, bigram, trigram, quadgram, and quintgram
frequency tables [8], one can design an English alphabet
predictor with a high success probability.

We have developed an English vocabulary typing predictor
where the successful predication probability can reach as high
as 24.73%, meaning almost 1 in every 4 letters. The following
chart shows the experimental results over texts of various types
such as literatures, emails, reports, magazines, and science
papers etc.

As for numeric inputs, it is even simpler. A string of
numeric inputs will be very likely followed by another digit
or a decimal point. This makes it even easier to predict and
inject bit-flipping errors.

**Example 3.1:** In a 128-bit AES-CBC protected system, the
16-byte IV is:

\[
\{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, \\
0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f\}
\]

And the 16-byte key is:

\[
\{0x60, 0x3d, 0xe0, 0x10, 0x15, 0xca, 0x71, 0xe8, \\
0x2b, 0x73, 0xae, 0xf0, 0x85, 0x7d, 0x77, 0x81\}
\]

When the victim is typing a sentence of “I owe you $1000”,
if the attacker predicts that “I owe you” will be followed by a
$ sign and an amount of money, he/she will be able to modify
the $’s cipher after “I owe you” by just one byte:

\[
\{0x71, 0xe8 \rightarrow 0xf0, 0x35, 0xf6, 0x88, 0x28, 0x6e, \\
0xc1, 0x3a, 0xd0, 0x87, 0x60, 0x10, 0x90, 0xd5, 0xe0\}
\]

which happens to makes $ decrypted to the “Pause” key
instead. More importantly, the following “1” will be decrypted
to “9” as shown in the plaintext’s scancode byte:

\[
\{0x00, 0x1e \rightarrow 0x26, 0x60, 0x3d, 0xe0, 0x10, 0x15, \\
0xca, 0x71, 0xe0, 0x2b, 0x73, 0xae, 0xf0, 0x85, 0x7d\}
\]

Now the entire message becomes: “I owe you 9000”.

**D. Replay**

The previous sub-section shows that with encryption only
the HID transmissions can be possibly tampered. However,
even if the user encrypts and authenticates every HID trans-
mission, if the attacker stores some legally encrypted ciphers
from the sender and replay them to spoof the user’s inputs, the
authentication process in the receiver will not be able to tell.
It is another potential threat to the current secure HIDs.

The AES-CBC mode has effectively randomized the plain-
texts during encryption so that the decrypted texts are be-
yond the control of attackers. However, since the keyboard
scancodes are limited to 8 bits as shown in Fig. 1 and out
of which there are at least 36 meaningful alphabetical and
numeric codes, it makes the replayed cipher from attackers to
be decrypted to another legal alphabetical or numeric scancode
with a probability of 36/2^8 which is at least 14%.

**Example 3.2:** In a 128-bit AES-CBC protected system with
the same parameters of the secret key and IV as **Example 3.1**, 
from the previous eavesdropping, the attacker has acquired a
legal cipher at the moment t_0 for a keystroke as:

\[
cipher(t_0) = \{0x17, 0x71, 0x98, 0x42, 0xac, 0x9c, 0x9e, \\
0xe8, 0x87, 0xc6, 0xed, 0x71, 0xd1, 0xe1, 0x78, 0x24\}
\]

And at the moment t_1 the victim typed “Please send Bob
from my bank account $1000” to his computer and then left
the table. The cipher for the last “0” is:

\[
cipher(t_1) = \{0xe0, 0x11, 0xe3, 0x4e, 0x23, 0xb1, 0x32, \\
0xf2, 0x4c, 0x12, 0x0a, 0x6d, 0x2c, 0x03, 0x87, 0xe1\}
\]

Then the attacker used his/her own dongle to send the
pre-stored cipher_{t_0} appending to the end of the text. By the
secret key and the previous cipher_{t_1} the decryption gets the
following plaintext under the CBC mode:

\[
plaintext(t_2) = \{0x00, 0x27, 0xe2, 0x41, 0xf2, 0x5f, \\
0x42, 0x07, 0x28, 0x59, 0x2a, 0x44, 0x52, 0xe2, 0x43, 0x5c\}
\]

where the scancode’s byte is \{0x27\} which happens to be “0”.
And the entire message now becomes: “Please send Bob from
my bank account $10000”.

The proposed secure scheme is designed to provide security
against all the MITM attacks illustrated above, to which the
current secure HIDs in the market are still vulnerable.

**IV. SYSTEM DESIGN CRITERIA**

The goals of the proposed security system are to protect the
HIDs from all the MITM attacks mentioned above while
keeping low transmission and power overhead. In addition, we
aim to modify the current secure HID scheme (128-bit AES-
CBC) as less as possible, so that it can be smoothly adopted
by the current HID manufacturers.
A. Against Eavesdropping

It should take the attacker hundreds of years to decrypt (understand) the user’s encrypted keystrokes or mouse movements and clicks by brute force without the security key. Thus the widely used 128-bit AES-CBC is applicable.

Therefore a more sophisticated randomization should be involved so that even with a record of ciphers and keystrokes, the attack is still not able to pair up the plaintexts and ciphers.

B. Against Hijack, and Tampering

Mouse hijack and keystroke tampering are different attacks but both take the advantage of current secure HIDs’ lack of authentication. The authentication procedure should ensure that even if the attacker applies active attacks in every transmission of keystroke of mouse movement/click, there will not be a single success in the device’s lifetime. Statistics have shown that the activity of the HID device for heavy users will not exceed 2^{31} times in a lifetime (24 hours a day, non-stop typing for 70 years). Therefore the attack mis-detection probability (attacks not detected / total number of attacks) should be at most 2^{-32}. To be on the safe side it should be at most 2^{-32}.

There are various ways for authentication. Keyed-hash Message Authentication Code (HMAC) provides strong security but requires a huge amount of extra bits for the digest, which is probably an over-kill and will bring considerable modification to the 128-bit AES-CBC mode on current HIDs. Here we propose to use the Algebraic Manipulation Detection (AMD) codes which is a keyless message authentication code [9]. Unlike HMAC which has a fixed length over 160 bits, it is very flexible to work on different sizes of blocks such as 32 bits. Its security level is determined by the size of the blocks. Moreover, the AMD codes brings in a random vector in encoding so that the plaintext is randomized even the blocks. Moreover, even if the secret key is leaked, as long as the attack is still not able to pair up the plaintexts and ciphers.

C. Against Replay

Each wirelessly transmitted keystrokes and mouse movement and click should be valid only once in its lifespan, so that even if the attacker stores all the authenticated and encrypted ciphers, he/she will not be able to reuse any of them. Thus it is necessary to use a timestamp in each transmission for the authentication process. The system always keeps track of the highest timestamp. If an incoming message has a timestamp smaller or equal to the highest one recorded, it is illegitimate.

D. Against Random Errors

Random errors are not attacks. They are usually caused by unstable transmissions or minor change of voltages etc. Most keyboards provide only error detection by parity check. We will add more reliability by applying error control codes (ECC) to the plaintexts. Since for this design the encoding of AMD codes works over $GF(2^{32})$ finite field, there are 16 bits left for the keystroke information part, which is more than enough for double-error correction. This provides much stronger reliability than the current keyboards.

E. Energy Consumption

Since wireless HIDs are mostly battery powered, the design should also aim for low energy consumption comparing with other possible methods.

Based on the above design criteria, we will propose the infrastructure and the parameters of the secure HID system, followed by the corresponding experiments and evaluation.

V. System Diagram and Workflow

The proposed protection scheme uses authenticated encryption with timestamps. Its encoding procedure is MAC-then-Encrypt (MtE). In this way the HID’s information portion (keystrokes and mouse clicks/movements), the timestamp, and the authentication signature can be all wrapped under 128 bits as the plaintext, adding no extra transmission overhead to the current HIDs equipped with 128-bit AES-CBC. Although MtE is not considered as the most generically secure in all authenticated encryption modes, it has been proved to be sufficiently secure with the AES-CBC mode [10].

A. Notations

To facilitate the description and evaluation of this protection scheme, we introduce the following notations:

- $GF$: the Galois finite field;
- $b$: the number of bits in each block;
- $\cdot$: the multiplication in the $GF(2^b)$ finite field;
- $\oplus$: the addition in $GF(2^b)$, namely bitwise XORs;
- $\oplus$: the accumulated sum in $GF(2^b)$;
- $\|$: concatenation;
- $k$: the information portion of the keystroke or mouse movement/click;
- $r$: the ECC redundancy to protect $k$ from random errors;
- $y$: the encoded information portion: $y = k|\cdot r$;
- $e$: the ECC’s syndrome for error correction;
- $s$: the ECC’s syndrome for error correction;
- $i$: the self-incrementing timestamps;
- $x$: the random vector;
- $e$: the AMD code’s signature;
- $P_{Miss}$: the attack mis-detection probability.

B. AMD Codes

The AMD codes have been known as a class of attack detecting codes against strong attacks, where attackers have the knowledge of the information portion (or a proper predication of it), the encoding scheme, and are able to apply malicious modifications to the transmitted messages. It often cooperates with cryptographic systems as a keyless authentication code [11]. Because of its random vector $x$, AMD codes performs excellently with equal security level under non-uniform distribution of the information portion, which befits the HIDs’ non-equilprobable inputs very well.

Definition 5.1: Let the $tb$-bit information portion $y = \{y_1, y_2, \cdots, y_t\}$, and the $tb$-bit random variable $x = \{x_1, x_2, \cdots, x_t\}$, the AMD code is constructed by [12]:

$$
\omega = \bigoplus_{j=1}^{t} (x_j \cdot y_j \oplus x_j^3); \quad \omega, x_j, y_j \in GF(2^b).
$$

If the error $e_y$ on $y$ is non-zero, then the term $x_j^3$ can be omitted. For the proposed protection scheme, $t = 1, b = 32$, $y = k|\cdot r$ which can be robustly combined with the timestamp $i$ by $y \cdot i$ [13], where $\cdot$ is finite field multiplication. The signature $\omega$ of the AMD codes is then computed by:

$$
\omega = y \cdot i \cdot x = (k|\cdot r) \cdot i \cdot x; \quad i, x, y \in GF(2^{32}).
$$
If the injected attack to each component is represented as 
\( e = \{ e_\omega, e_y, e_i, e_x \} \), the error masking equation will be:

\[
\omega + e_\omega = [(k||r) \oplus e_y] \cdot (i \oplus e_i) \cdot (x \oplus e_x). 
\]  

(3)

It has been verified that the right-hand side of the equation is always a non-zero polynomial of \( x \) of degree 1. It is easy to prove that for a certain message and an error \( e \), the error missing probability for this case is at most:

\[
P_{\text{miss}} = 2^{-b} = 2^{-32}. 
\]  

(4)

C. Error Correction Codes for Random Errors

Current keyboards apply parity check to detect odd number of random distortions on the scancodes caused by hardware faults, unstable voltages, or flaws in the transmission channels etc. Since the proposed scheme is encoded in 32-bit blocks and the keystroke information is only 16 bits, the rest of the 16 bits can be utilized by random error correction against double errors. This adds much greater reliability than the current solution in the market.

To ensure fast decoding and low hardware complexity, we propose to use the Orthogonal Latin Square Codes (OLSCs) [14]. For this case the error correction procedure is:

\[
H \cdot (\tilde{k}||\tilde{r}) = S 
\]

(5)

\( \tilde{k} \in GF(2^{16}) \) and \( \tilde{r} \in GF(2^{16}) \) are distorted keystroke information and redundancy portions. \( H \) is a 16 \( \times \) 32 binary OLS matrix, and \( S \) is the 16-bit binary vector used for one-step majority voting to correct up to 2 random errors [15].

As for the mice, since its information part takes up already 32 bits and its movement displacements tolerate minor errors, it is not compulsory to have this feature.

D. System Diagram

As stated in the beginning of this section, the proposed scheme is structured by authenticated encryption with MAC-then-Encrypt work flow. The ECC’s redundancy \( r \) protects the information portion \( k \) from up to 2 random errors. The timestamp \( i \) will guarantee that each transmitted message can never be replayed again. The random vector \( x \) randomizes the plaintext \( ((k||r)||i||x||\omega) \) so that the attackers cannot acquire the mapping between the HID inputs and ciphers even if they have access to both of them as the preconditions 1) and 2) [16]. The AMD authenticating signature \( \omega \) verifies if the message has been maliciously altered or not. And in the end the AES-CBC encryption process will protect the system against eavesdropping.

As shown in the system diagram, the proposed secure system firstly encodes the 16-bit keystroke \( k \) into the 32-bit information part \( y \), with a double-error correcting OLSC code (for mice \( k = y \in GF(2^{32}) \)). Secondly they are encoded with the timestamp \( i \) and the random vector \( x \) into \( \omega \) by the message authentication code AMD. Then the 128-bit \(( (k||r)||i||x||\omega) \) will serve as the randomized plaintext to be encrypted by the 128-bit AES-CBC module. The Man-In-The-Middle attacks occur in the wireless transmission channel by eavesdropping, tampering, and replay. On the receiver end, the AES firstly decrypts the plaintext and then the first 32 bits are sent to the OLSC decoder for random error correction. Then it checks if the timestamp \( i \) has incremented or not. Finally the entire plaintext is verified by the AMD decoder for attack detection.

Fig. 5: The flow chart of the proposed scheme’s encoding and decoding procedures. It does not only provide security against MITM attacks, but also more reliability against random errors.

VI. EVALUATIONS

In this section we will evaluate the proposed scheme’s security level by \( P_{\text{miss}} \), and energy consumption by comparing the proposed scheme with other possible schemes.

A. Attack Mis-detection Probability

The attack mis-detection probability is calculated by attacks not detected divided by total number of injected attacks. We have run through tests on more than 1 billion HID inputs from 16 types of texts including literatures, emails, magazines, news, and science papers etc. During the simulation the system mimics an user typing on a keyboard and using a mouse, and an attacker applying different active MITM attacks in every transmission, while the receiver trying to detect the attacks.

Since the 32-bit system provides so strong security that not a single attack was missed in the tests, we apply various sizes of data blocks (from 4 to 32 bits) due to AMD codes’ flexibility to observe how much the experimental attack mis-detection probability matches \( P_{\text{miss}} = 2^{-b} \) in (4).

TABLE I: \( P_{\text{miss}} \) under 1,030,779,042 Active MITM Attacks

<table>
<thead>
<tr>
<th>Missed</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missed Errors in Experiments</td>
<td>63,741,469</td>
<td>4,031,649</td>
<td>15,760</td>
<td>0</td>
</tr>
<tr>
<td>Experimental ( P_{\text{miss}} )</td>
<td>6.18e-2</td>
<td>3.91e-3</td>
<td>1.53e-5</td>
<td>0</td>
</tr>
<tr>
<td>Theoretical ( P_{\text{miss}} = 2^{-b} )</td>
<td>6.25e-2</td>
<td>3.91e-3</td>
<td>1.53e-5</td>
<td>2.33e-10</td>
</tr>
</tbody>
</table>

\(^1\) Under more than 1 billion active MITM attacks, the proposed scheme caught all the replay attacks, and missed tampering and hijack attacks with a probability very close to the theoretical estimation \( 2^{-b} \).
The experimental result not only shows that the proposed protection scheme works according to the theoretical estimation of $2^{-b}$ error mis-detection probability precisely, but also demonstrates that the 32-bit scheme is secure enough for missing $0$ attack under 1 billion maliciously modified HID inputs, which is more than most people’s lifetime HID usage.

**B. Transmission and Power Consumption Overhead**

As mentioned in IV. B, another approach is AES + HMAC + timestamps where HMAC provides authentication. However the HMAC based scheme requires at least 160 bits to provide $2^{-80}$ attack mis-detection probability, which is an overkill to the security required and brings too much modification to the existing 128-bit AES secured HID systems.

As for the 32-bit AMD code and timestamp based scheme, since all computations are done in the 32-bit finite field, it saves largely the transmission overhead, hardware area, and energy consumption over the HMAC authentication method. Even if the scheme upgrades $x$ and $\omega$ to 80-bit mode to achieve the same $P_{\text{miss}}$ as the HMAC based scheme, it still saves tremendously the energy on computing the signature. The following comparison was made based on the implementation on Xilinx Vertex 4 FPGA and Cadence SOC Encounter:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$P_{\text{miss}}$</th>
<th>Extra Bits Over AES</th>
<th>Area (um$^2$)</th>
<th>Energy (nJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Scheme (32 bits)</td>
<td>$2^{-32}$</td>
<td>0</td>
<td>3093.6</td>
<td>2.10</td>
</tr>
<tr>
<td>Proposed Scheme (80 bits)</td>
<td>$2^{-80}$</td>
<td>128</td>
<td>6274.8</td>
<td>7.49</td>
</tr>
<tr>
<td>HMAC Based (160 bits)</td>
<td>$2^{-80}$</td>
<td>128</td>
<td>58813.7</td>
<td>58.06</td>
</tr>
</tbody>
</table>

\*1 The 32-bit proposed schemes requires 0 extra bits in addition to the existing secure HID protocol, while the other 2 need an extra 128-bit AES block.

\*2 Both the 32 and 80-bit proposed schemes take around 1/10 of the HMAC based scheme’s hardware area.

Compared to the HMAC (low-power implementation) base scheme, the 32 and 80-bit proposed schemes even consume much less. The 30-bit proposed scheme is only 3.6% of the HMAC’s and the 80-bit scheme 12.9% under the same security level. This advantage becomes more prominent if the protected devices are powered by batteries.

**VII. CONCLUSION**

This design is proposed under the motivation of the existing and potential Man-In-The-Middle attacks to the secure wireless keyboards and mice in the market. We have proved by theory and experiments that by authenticated encryption with timestamps, it provides sufficient security against MITM attacks by missing not a single error in a device’s lifespan. Moreover, its power consumption is less than 13% of other possible solutions. Thus it can be applied to other power-sensitive devices such as IoT devices with limited hardware and battery resources, and defibrillators whose battery replacement takes a surgery, as in recent years they have become targets of attackers due to their insecure wireless transmissions [17].

**REFERENCES**


