Using IIO Structure to enable additional workflows

Margaret Sturgill, Steven J Simske, Marie Vans; HP Labs, Fort Collins, CO, USA

Abstract

Serialization is an important VDP (variable data printing) application. Incremental Information Objects (IIOs) allow us to keep track of steps in a workflow by modifying their serialized information at each step. Usually the IIO progression is defined by specific rules that change the visual properties of the IIO. We herein propose an additional constraint that defines a specific structure to the underlying bit string defining the IIO. We can then use the structure to either identify the workflow stage without access to the main ID database; modify the current workflow stage due to additional factors (user authentication, location, etc.); or use the information in the IIO as additional security information

Problem

Incremental Information Objects (IIOs), such as progressive barcodes [1,2,3,7], provide us with a multicolored printed mark, wherein the colors change at each stage in accordance to a specific rule. In general, at each stage of the progression we can obtain the binary string encoded, but the actual stage is not known without a central database lookup to verify the IIO data. In many cases, the access to such a database may not be available, so we were interested to see if it is possible to impose additional constraints on the binary representation of each IIO allowing us to infer additional information about the stage of the IIO in the workflow. In this way, the mark carries not only *explicit data* (actual data stored in the IIO) but also *implicit data* (data based on the structure of the data in the IIO). We wish to trigger a variety of workflows at each stage without the knowledge of the meaning of the data stored in the IIO.

IIOs can come in two types, and we would like our solution to be usable in both cases:

1. One-to-One stage correspondence where each stage transition to a unique string with a specific number of set bits combined with additional (end-user information, workflow menus, expected set of options etc.) triggers a specific workflow. See Figure 1. In this case we can randomly pick any transition that will generate a new IIO that supports correct number of bits in the next transition. The colors progress in White->CMY->RGB sequence
2. Many-to-One stage correspondence. Where a single IIO can transition to multiple possible IIOS (item-container relationship). The number of set bits combined with the user's authentication level can trigger workflows on child items even when the end-user has no visibility to the container contents. See. Figure 2. In Many-To-One IIOs, the visual progression works opposite to One-to-One progression. It progresses in RGB->CMY->White direction.

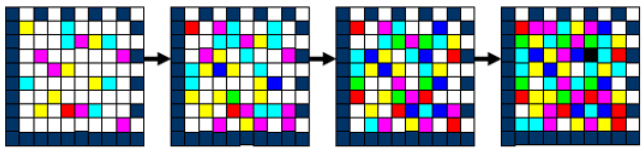

Figure 2. One-to-One IIO progression

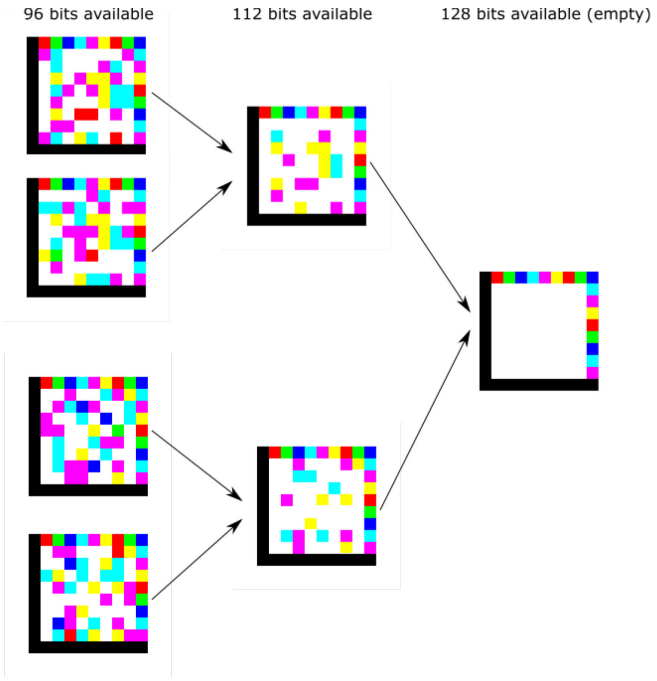

Figure 1. One-to-Many Progression

Proposed Solution

As our solution, we propose a structure at each stage of the IIO that guarantees a specific number of 1-bits (with the remainder being 0 bits, of course) in the data string. For example, we might require that for an IIO containing 128 bits of data, we impose a structure such that:

Stage 0 (0 bits as binary "1") -> Stage 1 (32 bits as binary "1") -> Stage 2 (64 bits as binary "1") -> Stage 3 (96 bits as binary "1") -> Stage 4 (All 128 bits as binary "1")

Usually we will ignore Stage 0 and Stage 5 as there is no entropy in either one of the stages. This is due to the fact that at Stage 0 there are no available bits (the IIO is potentially full) and at Stage 4 all 128 bits are available thus not carrying any information (blank IIO in Figure 2).

In above case, we can ascertain the current stage by counting the number of bits available for transition in the IIO and then

inferring the progression stage from the result. Once the progression stage is known, specific workflows can be triggered without the user having access to the main database.

These workflows would be selected based on other information available at the time such as user permissions, location or time of the day.

Example of Application to Supply Chain:

The two different types of progression could be used for different supply chain application.

Many-to-One transition in shipping applications:

For example, let us take an IIO for which each stage indicates a level of parent-child relationship in a shipping application. Additionally, we have two users with varying degrees of privileges to open the packaging. The IIOs identifying the container have the following structure imposed (number of 1 bits at each of the stages):

Shipping Container (128) -> Pallet (96) -> Box (64) -> Item (32)

We can then trigger different workflows based both on the structure of the IIO and the permission level of the user scanning the IIO:

Let us consider a role-based access control example. Here we have 160 bits of writable string. As previously mentioned, due to lack of entropy at 0 and 160 bits, we will only consider at most 128 bits of information. If User 1 (full privileges) scans a Box (64 set bits in the IIO) the system might automatically record the fact that deep inspection of the container is occurring. If User 2 (limited privileges) scans the same Box, the system might trigger an investigation as to why user 2 had access to the Shipping Container contents. The system might not know what data is stored in the IIO, but by knowing the stage and the user privilege level different workflows can be triggered. Note that the privilege level can be asserted in different ways. In some cases, the possession of the reader might automatically grant very limited access privileges, knowledge of login information like password a mid-level privileges, while biometric information might be necessary to grant full access privileges. The solution is safe from data mining and other "temporal attacks" since a rules engine can dictate whether a given progression stage is currently available. That is, a fraudulent agent may try to "rewind" privileges from Stage 3 to Stage 2 if she has less privileges at Stage 3 (revocation); however, the back-end rules engine is aware that Stage 2 privileges are no longer available.

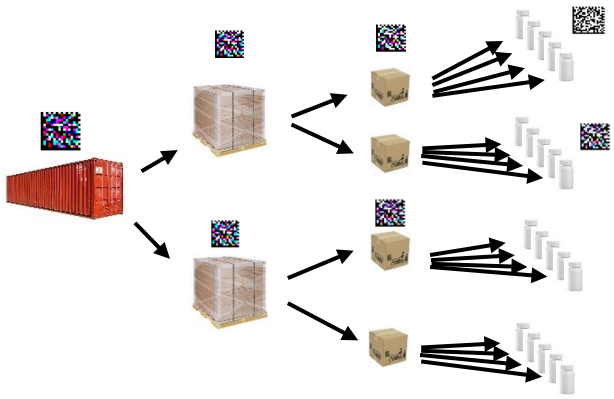

Figure 3. Application of IIO to a transportation use case.

Table 1. Sample workflow triggers in a Many-to-One use case

Number of bits in IIO	User 1 Full privileges	User 2 Limited privileges
128 *Shipping container*	Record item location	Record item location
96 *Pallet*	Record cursory inspection	Record item location
64 *Box*	Flag possible problem	Record item location Triger investigation
32 *Individual item*	Record deep inspection	Record item location Triger investigation

Example of One-To-One application in data retrieval:

A One-to-One progression IIO can be used to track an individual workflow with the structure of the IIO changing at each stage (of a 192-bit serialization field):

Stage1 (160 bits) -> Stage 2 (128 bits) -> Stage 3 (96 bits) -> Stage 4 (64 bits)->Stage 5(32 bits)

Let us have three Users with different permission levels. At each stage, each of the Users have a different permission to access the data. For example, let us examine a data retrieval request where the system can automatically limit an access to data even when each of the users that can see that the request has been notified of the requests existence. User 1 can create request, User 2 can route that request to the correct recipient and create a resolution, and User 3 can process the data.

By examining the structure of the IIO combined with the nature of the user credential permission, the system does not need to know anything about the content of the data or even access the data repository. This limits the possibility of the data being accessed by unauthorized parties

Table 2. Data permission in One-ot-One data routing case.

Number of bits in IIO	User 1	User 2	User 3
160	Enter request	Has no access to request content or data.	Has no access to request content or data.

128	View initial request	Can update the request and route to User 3. No access to data	Has no access to request content or data.
96	View initial request	View updated request	Access the request and data and route back to User 3
64	View initial request	Receive data from User 3 trigger additional workflow to process the return data. Create resolution	Has no access to request content or data.
32	View initial request and resolution	Has no access to request data or resolution.	Has no access to request data or resolution.

Example of Application to Security Printing:

The above examples use the structure of the IIO to trigger specific workflows without accessing the data associated with the unique ID stored in the IIO. While usually the IIO data at each stage carries an identifier data, we could instead assume that some of the stages carry some kind of security data. If, for example, a payload string *s* will be decrypted via an XOR operation (signified by the symbol ⊕) with a known nonce (one-time use random string) *n*, we may want to make sure that the decryption will occur only once a certain stage (with data string *a*) has been reached (or later) in the progression. Then, we can require that the decryption is performed by*: a⊕n⊕s.* Note that if *s* availability stage is reached before the nonce is available, we can pre-process string *s* by calculating *a⊕s*. Again the structure of the IIO data would indicate the stage and thus the specific strings *a, n* and *s* needed for decryption. The approach allows the set of approvals (each of which performs an XOR operation on the encrypted string) to be performed in any order. See Tables 3 and 4.

Note that in this example the security data must be the same length at each stage. Thus, the amount of data that can be carried at any stage is limited by IIO carrying the least information.

Let us consider a progression example where 3 of the progression stages indicate:

Stage A – Decryption Authorization Stage carrying an activator
Stage N – Decryptor Stage carrying the nonce
Stage S – Data Stage carrying the encrypted data string

The IIOs encoding these stages may be encountered in any order, but all three have to be read before the decryption can be performed. This is enforced by the fact that the data in all three IIOs has to be combined to perform the decryption.

Table 3 shows what happens when the progression stages are read in the Stage A-> Stage N -> Stage S order. That is the authorization is obtained before the data is available. In this situation, the nonce can be pre-processed with the authorization *a⊕n* generating a new intermediate string *d*. When Stage S is reached the decryption can be finished by performing *d⊕s*

Table 4 shows the case when the nonce *n* and security data *s* are available before the decryption authorization is obtained. In this case we can process string *s* with the nonce *n* to obtain the string *d*. Once the authorization stage is reached, the decryption can be finalized by combining the string *d* with the authorization

Even though the XOR operations were performed in different order, the final result is the same in both examples.

Table 3. Decryption authorization stage *a* is available before the data *s* is available. Nonce *n* can be authorized before the data is available, and *a⊕n⊕s* is the orders of XOR operations.

	Binary string
a – activator from IIO	1 0 1 0 1 1 0 1 0
n – nonce	0 1 1 0 1 1 0 0 1
s – data code	0 0 1 0 0 1 0 1 1
d = a⊕n	1 1 0 0 0 0 0 1 1
d⊕s	1 1 1 0 0 1 0 0 0

Table 4. Nonce *n* and string *s* are available before we reach the decryption authorization stage *a*. In this case *n⊕s⊕a* can be processed at decryption authorization stage.

	Binary string
a – activator from IIO	1 0 1 0 1 1 0 1 0
n – nonce	0 1 1 0 1 1 0 0 1
s – data code	0 0 1 0 0 1 0 1 1
d = n⊕s	0 1 0 0 1 0 0 1 0
d⊕a	1 1 1 0 0 1 0 0 0

As mentioned before, because each stage IIO carries a different length data string, it is necessary to pad the payload string to the appropriate length. Even though we can use random strings to fill in the additional information, care should be taken to prevent "entropy snooping" and allow a possible reverse engineering approval, order, or the workflow in general. In the case of the example above, note that the Hamming Distance [8] (number of bits that vary between the two strings) is the same between *a* and *s* strings and *n* and *s* strings (hamming distance of 3); thus, it is harder to reverse-engineer which stage carry the authorization vs. the nonce.

Conclusions

While IIOs allow us to encode *explicit* data that changes over time, we can also provide a secondary channel of information that is stage-specific. Rather than the actual data stored in the IIO, it is its structure that triggers the workflow. By separating the two channels, we liberate the processing system from the necessity of looking up the meaning of the IIO content, thereby minimizing the level of privileges needed to process IIO data.

References

[1] S J. Simske, A.M. Vans and B. Loucks. 2012. Incremental Information Objects and Progressive Barcodes, in Proceedings of the

Digital Fabrication and Digital Printing Conference, NIP28, Quebec City, Quebec, Canada, pp. 375-377.

[2] Simske SJ, Vans, M. Archive-enabling Tagging using Progressive Barcodes. In Proceedings of the Archiving'15 Conference, May 19-22, 2015, Los Angeles, pp. 130-135.

[3] Simske S, Vans M, Pollard S, Adams G, "Forensic Markings for Progressive Barcodes." TAGA 67th Annual Technical Conference, March 23, 2015, Albuquerque.

[4] Simske SJ, Vans M, "Applications for Progressive Barcodes", Journal of Imaging Science and Technology (July 2014), 58(4), Article no. 40404 (9 pp.). DOI: 10.2352/J.ImagingSci.Technol.2014.58.4.040404

[5] Simske SJ, Vans M, Loucks B, "Incremental Information Objects and Progressive Barcodes," Journal of Imaging Science and Technology (May 2013), 57(3), Article no. 030405. DOI: 10.2352/J.ImagingSci.Technol.2013.57.3.03040

[6] A.M. Vans, S J. Simske, and B .Loucks. 2012. "Progressive Barcodes", in Proceedings of the Digital Fabrication and Digital Printing Conference, NIP28, Quebec City, Quebec, Canada, pp. 368-370.

[7] Vans, Marie, Steven Simske, and Brad Loucks. "Progressive Barcode Applications." NIP & Digital Fabrication Conference. Vol. 2013. No. 1. Society for Imaging Science and Technology, 2013.

[8] Hamming, R. W. Error detecting and error correcting codes. Bell System Tech. J. 29, (1950). 147–160.

Author Biography

Margaret Sturgill currently works at HP Labs in Fort Collins, Colorado in the HP Labs Print Adjacencies & 3D Lab. Her main interests include Document Security, Document Workflows, Supply Chain Analysis and Anti-counterfeiting. She holds a BS in Computer Science and Mathematics from University of Kentucky and a Ph.D. in Computer Science from University of Utah. She has previously worked at HP on scanner image processing software and at Ataman Software Inc as the Chief Operating Officer. She has 25 US Patents.